

Exhibit 2

**IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION**

SFA SYSTEMS, LLC f/k/a
TRITON IP, LLC

v.

INFOR GLOBAL SOLUTIONS
(MICHIGAN), INC., et al.

§
§
§
§
§
§
§

CIVIL ACTION NO. 6:07-cv-067[LED]

JURY

**INFOR GLOBAL SOLUTIONS (MICHIGAN), INC.’S
FOURTH AMENDED INVALIDITY CONTENTIONS**

Pursuant to Patent Rule 3-3 and 3-6, Defendant Infor Global Solutions (Michigan), Inc. (“Infor”) submits its Amended Invalidity Contentions. Infor’s investigation regarding prior art grounds of invalidity is on-going and Infor reserves all rights to further supplement or amend its invalidity contentions as permitted by the Patent Rules.

I. INDEFINITENESS, NON-ENABLEMENT, OR LACK OF WRITTEN DESCRIPTION UNDER 35 U.S.C. § 112(1) OR (2)

Pursuant to Patent Rule 3-3(d), Infor contends that the asserted claims of the ‘525 patent are invalid under 35 U.S.C. § 112, ¶ 1 because the specification does not enable and/or does not have adequate written description of many of the claim limitations. The patent specification would not have enabled one of ordinary skill in the art to practice the claimed inventions of the ‘525 patent without undue experimentation.

The written description requirement of 35 U.S.C. § 112 was not met by elements/steps recited in the below-described claims because the claims contain subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

The enablement requirement of 35 U.S.C. § 112 was not met because the claims contain subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The disclosure provides no teachings related to the claimed features discussed below. A person of ordinary skill in the art would be unable to make and/or use the claimed apparatus because there is no explanation of its design, structure, or functionality beyond the claim language. The claim language itself defines the invention and does not provide enabling support for the functionality of the features.

Claims 13-19 and 24-39 were added after the filing of the original application, and of these at least the following claims include terms that fail to satisfy the written description or enablement requirements of 35 U.S.C. § 112.

A. No Written Description or Enablement of the Term “Inferred Event Distributed Among at Least Two of the Plurality of Subsystems”

Claims 14 & 26: The specification provides no description of an inferred event distributed among at least two of the plurality of subsystems; nor does the specification provide a description how the inferred event would be distributed among at least two of the plurality of subsystems.

Claims 15 & 27: Fail the written description and enablement requirements due to their dependency on claims 14 & 26, respectively.

Claims 16 & 28: Fail the written description and enablement requirements due to their dependency on claims 15 & 27, respectively.

B. No Written Description or Enablement of the Term “Inferred Context”

Claims 16 & 28: The specification provides no description of an inferred context including an identification of the subsystems among which the inferred event is distributed; nor

does the specification provide a description of how or where the inferred context would be identified.

Claims 19 & 31: The specification provides no description of an inferred context including an identification of the subsystems among which the inferred event is contained; nor does the specification provide a description of how or where the inferred context would be identified.

C. No Written Description or Enablement of the Method Step “Inferring Occurrence of an Event While _____”

Claims 33 through 39: The specification provides no description of a scenario where an occurrence of an event is inferred “while” converting a name to a potential customer; “while” converting a lead to a buying customer; “while” converting a lead to a buying customer and prompting the buying customer to make a buying decision; etc., as recited in the respective claim. The specification fails to provide a significance of inferring the occurrence of an event “while” doing something else, such as converting a name to a potential customer (as in claim 33), including describing whether the converting a name to a potential customer is recorded as being associated with the event that occurred, whether the converting a name to a potential customer might be the event that occurred, or whether the converting a name to a potential customer might be the context in which the event occurred. Accordingly, the specification does not support claims 33-39.

D. No Written Description or Enablement for Terms In Independent Claims 1, 20, and 40

Independent claims 1, 20, and 40 were amended during prosecution adding new matter that is neither described nor enabled. Accordingly, the following claim terms added by amendment fail the written description and/or enablement requirement of 35 U.S.C. § 112¶ 1.

E. No Written Description or Enablement of the Terms “Infer,” “Inferring,” “Inferring Occurrence of the Event and a Context in Which the Event Occurred,” or “Infer Occurrence of the Event and a Context in Which the Event Occurred”

Claims 1 & 20: the element: “determining a context in which the event occurred” was deleted in view of prior art and new matter, “inferring occurrence of the event and a context in which the event occurred” neither disclosed, described, nor enabled was added.

Claim 40: the element: “detect the occurrence of a first event in the sales process” was deleted in view of prior art and new matter, “infer occurrence of the event and a context in which the event occurred...” neither disclosed, described, nor enabled was added.

The term infer/inferring does not exist in the specification. The original disclosure is completely lacking of any description of an embodiment where the occurrence of an event and a context in which the event occurred is inferred. All limitations must appear in the specification. The failure to expressly describe the inferring of an occurrence of the event and a context in which the event occurred results in a specification that fails to convey to one of ordinary skill that the inventors were in possession of the invention. Further, the specification fails to enable one skilled in the art how to make and/or use the invention. The disclosure provides no teachings related to the inferring of an occurrence of the event and a context in which the event occurred. A person of ordinary skill would be unable to make and/or use the claimed apparatus because there is no explanation of its design, structure, or functionality beyond the claim language.

F. No Written Description or Enablement for the Term “Changes in State Characteristic”

Claim 1: the element: “recognizing an event” was deleted in view of prior art and new matter, “detecting one or more changes in state characteristic of an event occurring within the system,” neither disclosed, described, nor enabled was added.

Claim 20: the element: “automatically detecting the occurrence of the first event” was deleted in view of prior art and new matter, “automatically detecting one or more changes in state characteristic of an event occurring in the sales process” neither disclosed, described, nor enabled was added.

Claim 40: a new element, “detect one or more changes in state characteristic of an event occurring in the system” neither disclosed, described, nor enabled was added.

The phrase “changes in state characteristic” does not exist in the specification. The original disclosure is completely lacking of any description of what is a change in state characteristic. All limitations must appear in the specification. The failure to expressly describe a change in state characteristic results in a specification that fails to convey to one of ordinary skill that the inventors were in possession of the invention. Further, the specification fails to enable one skilled in the art how to make and/or use the invention. The disclosure provides no teachings related to what constitutes a change of state characteristic, when or at what threshold would a change of state characteristic of an event be satisfied, and therefore recorded or detected, and whether a change of state characteristic means that an event has occurred and is complete. Further, would inferring the occurrence of the event mean that the event has occurred and is complete, or merely that a change of state characteristic of the event has been detected. Accordingly, from the original specification a person of ordinary skill would be unable to make and/or use the claimed method and/or system because there is no explanation of its design, structure, or functionality beyond the claim language.

G. No Written Description or Enablement for the Term “Expert System”

Claims 41 and 42: the element “expert system,” as that term is understood to one of skill in the art is neither disclosed, described, nor enabled in the specification.

II. PRIOR ART REFERENCES

Pursuant to Patent Rule 3-3(a) and (b), Infor identifies each of the following items of prior art that anticipates the claims of the '525 patent under 35 U.S.C. § 102:

1. Vernon, P., "Mott's Business-Management Expert Planning System," First International Conference on Expert Planning Systems, 1991, June 27-29, 1990;
2. U.S. Pat. No. 4,853,852, issued August 1, 1989;
3. U.S. Pat. No. 5,450,314, issued September 12, 1995;
4. U.S. Pat. No. 4,567,359, issued June 28, 1986;
5. U.S. Pat. No. 5,216,592, issued June 1, 1993;
6. U.S. Pat. No. 5,283,856, issued February 1, 1994;
7. GoldMine Software v. 2.5a, first publicly used, sold, and offered for sale at least as early as October 1994 by GoldMine Software Corporation;
8. GoldMine Software v. 2.5a Reference Manual, 1994 release;
9. GoldMine Factsback v. 2.5a Bulletin #371, Nov. 1993;
10. Casanova, et al., "Expert System for Automatic Authorization of Deficits." IEEE, 1989;
11. Unnamed software developed by Jerome Johnson and/or Clear With Computers beginning in 1983 and all changes, revisions, versions, iterations, or embodiments thereof from 1983 through and including October 30, 1994 as sold to Clear With Computers' clients such as, for example, International Harvester, General Motors, Freightliner, PACCAR, and/or IBM("the Johnson Software");
12. U.S. Pat. No. 5,774,868, filed on Dec. 23, 1994, and issued Jun 30, 1998;
13. .U.S. Pat. No. 5,201,010, issued Apr. 6, 1993;
14. U.S. Pat. No. 5,349,662, issued Sept. 20, 1994;
15. U.S. Pat. No. 5,283,865, issued Feb. 1, 1994;
16. U.S. Pat. No. 5,367,627, issued Nov. 22, 1994;
17. U.S. Pat. No. 5,717,595, issued Feb. 10, 1998;
18. U.S. Pat. No. 6,023,683, issued Feb. 8, 2000;

19. U.S. Pat. No. 6,061,506, issued May 9, 2000;
20. U.S. Pat. No. 5,630,127, issued May 13, 1997;
21. U.S. Pat. No. 5,657,233, issued Aug. 12, 1997;
22. U.S. Pat. No. 4,931,932, issued June 5, 1990;
23. U.S. Pat. No. 5,309,355, issued May 3, 1994;
24. U.S. Pat. No. 5,191,522, issued Mar. 2, 1993;
25. U.S. Pat. No. 5,446,653, issued Aug. 29, 1995;
26. U.S. Pat. No. 5,168,445, issued Dec. 1, 1992;
27. Harrison, H.C. & Qizhong, Gong: "An Intelligent Business Forecasting System,"
Association of Computer Machinery 089791-558-5, 1993;
28. Stone, Robert W. & Good, David J.: "Expert Systems and Sales Strategies,"
Association of Computer Machinery 089791-416-3, 1990;
29. U.S. Pat. No. 5,347,632, issued Sep. 13, 1994;
30. U.S. Pat. No. 4,947,028, issued Aug. 7, 1990;
31. U.S. Pat. No. 5,117,354, issued May 26, 1992; and

To the extent SFA contends that the references cited above do not anticipate each and every limitation of the claims of the '525 patent, Infor asserts that any limitations omitted from a respective single reference would have been obvious to one of skill in the art, and/or are disclosed, taught or suggested in another of the above-identified references. Accordingly, Infor contends that the above-identified prior art also renders the claims of the '525 patent invalid under 35 U.S.C. § 103.

Motivation to combine any of the above-identified references exists as all are analogous art, and since the respective combination would enhance the performance of any sales oriented, or business directed, hardware and/or software system. For example, it would have been obvious to add an event manager to any sales oriented, or business directed, hardware and/or software

system so that user control of the entire system could occur from a single management person or location, and/or so that additional system components could be integrated into a single overall system to enhance overall system performance. As a further example, it would have been obvious to direct a business oriented, event-driven intelligent system, capable of considering events and related information to make informed suggestions, to a sales and marketing purpose, since doing so could enhance sales performance just as the system previously enhanced related general business performance.

In addition, pursuant to Patent Rule 3-3(a) and (b), Infor identifies each of the following items of prior art that render the claims of the '525 patent obvious under 35 U.S.C. § 103:

1. Spezialetti, Madalene: "An Approach to Reducing Delays in Recognizing Distributed Event Occurrences," Association of Computer Machinery 0-89791-457-0/91/0011/0155, 1991.

In addition, Infor identifies The Johnson Software as prior art under 35 U.S.C. § 102(b). The Johnson Software was publicly known, publicly used, offered for sale, and/or sold more than one year prior to the filing date of the '525 patent. For example, the Johnson Software was sold to International Harvester in or around 1983. In addition, the Johnson Software was to General Motors, Freightliner, PACCAR, and/or IBM between 1983 and October 30, 1994.

III. PRIOR ART CHARTS

Pursuant to Patent Rule 3-3(c), Infor attaches claim charts showing how one or more of the references listed above teaches elements of claims of the '525 patent. Infor's claim charts are subject to revision and amendment pursuant to Federal Rule of Civil Procedure Rule 26(e) and following discovery or the Court's construction of the claims at issue. To the extent that the following contentions reflect constructions of claim limitations consistent with or implicit in Plaintiff's infringement contentions, no inference is intended nor should any be drawn that Infor

agrees with Plaintiff's claim constructions, and Infor expressly reserves its right to contest such claim constructions. Infor offers such contentions solely in the alternative to any position it may ultimately take as to claim construction and non-infringement issues.

The attached claim charts show where each element of each asserted independent claim is found in each of the prior art references. Infor states that the additional limitations contained in each of the dependent claims do not add anything novel or non-obvious to the independent claim from which they depend. More specifically, the dependent claims fail to add any novel element and merely describe obvious and known substitutions of the claimed element of the independent claim from which they depend. Accordingly, it would have been obvious to one of skill in the art to use any or all of these obvious substitutions as the embodiment of the general limitation in a particular system.

IV. INEQUITABLE CONDUCT

Pursuant to Patent Rule 3-3(a) and (b), Infor discloses that the '525 patent is unenforceable due to inequitable conduct during the prosecution of the '525 patent. Specifically, the inventors, attorneys, and others involved in the prosecution of the '525 patent failed to disclose material information to the Patent Office during the prosecution of the '525 patent.

One of the named inventors of the '525 patent, Jerome Johnson, first made, used, sold, and offered for sale a software program, at least as early as 1983, that embodies the invention disclosed and claimed in the '525 patent. Mr. Johnson's software is prior art to the '525 patent, was material to the prosecution of the '525 patent, and was not disclosed to the USPTO by anyone involved in the prosecution of the '525 patent during the prosecution of the '525 patent. Mr. Johnson's software is not cumulative of any other information disclosed to the USPTO during the prosecution of the '525 patent. On information and belief, the failure to disclose Mr. Johnson's software to the USPTO during the prosecution of the '525 patent was done with the

intent to deceive the Patent Office. The intentional failure to disclose Mr. Johnson's software to the USPTO during the prosecution of the '525 patent constitutes inequitable conduct, which renders the '525 patent unenforceable.

V. DOCUMENT PRODUCTION

Pursuant to P. R. 3-4(a), Infor is making available for inspection and copying source code to show the operation of any aspects or elements of each accused instrumentality identified by SFA in its P. R. 3-1(c) chart.

Pursuant to P. R. 3-4(b), Infor is producing or making available for inspection and copying each item identified pursuant to P.R. 3-3(a) that does not appear in the file history of the patents at issue.

DATED: May 4, 2009

BLANK ROME LLP

By: /s/ Alfred W. Zaher
Alfred W. Zaher
Bruce D. George
Joel L. Dion
One Logan Square
Philadelphia, PA 19103
Phone: (215) 569-5364
Fax: (215) 832-5364
Attorneys for Defendant
Infor Global Solutions (Michigan), Inc.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a copy of the foregoing Fourth Amended Invalidity Contentions was served on counsel for defendant on this 4th day of May, 2009 by electronic mail to:

Andrew Wesley Spangler
Spangler Law PC
208 N. Green St.
Suite 300
Longview, TX 75601
903-753-9300
spangler@spanglerlawpc.com

John J. Edmonds
The Edmonds Law Firm, PC
709 Sabine Street
Houston, TX 77007
(713) 858-3320
johnedmonds@edmondslegal.com

David M. Pridham
Law Office of David Pridham
25 Linden Road
Barrington, Rhode Island 02806
(401) 633-7247
david@pridhamiplaw.com

/s/ Joel L. Dion
Joel L. Dion, Esquire

U.S. Pat. No. 6,067,525 Claim 1	Mott's Business-Management Expert Planning System	U.S. Pat. No. 4,853,852
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see §2.4 is a Sales-Programme Subsystem within an EPS and business management planning system (See Abstract). See ¶37: A tool to predict which prospects will materialize and how soon	The preamble is not a limitation. Nonetheless, see Abstract, FIG. 1 and col. 3, lines 4-11: A marketing tool having a portable lap computer for recording a sales call, and a master computer to automatically prepare planners
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Abstract & ¶3: four subsystems are resource scheduling, market-intelligence, technical expertise and sale programme. See ¶6: expert system includes knowledge database, an inference engine and information database. See ¶11 & FIG. 5: The Information database includes the four subsystems as well.	FIG. 1 and col. 3, lines 19-31: lap computer of each salesperson, each employer computer, the master computer system and all related databases are the subsystems.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	§ 4: Inference Engine is the event manager, coupled to knowledge and information databases with four subsystems.	FIG. 1 & col. 3, lines 19-24 – the master computer facility is the event manager, which detects the day's customer call information from each salesperson of each employer (each visit is an event).
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	¶39 & FIG. 10: Table of Prospect Attributes - ¶25: Market Intelligence. ¶27: In obtaining and analyzing market intelligence, trends about changing markets are identified, providing pointers to changes in technical-expertise subsystem. ¶40 & FIG. 11: Model's product range is event occurrence, context is prospect table of attributes.	Col. 4, lines 39-62: the master computer infers event occurrence (customer visit) and a context of the event (the employer of each salesperson).
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	¶40 & FIG. 11: A more comprehensive product package is developed from event occurrence and attribute information. ¶44: Inference engine uses and applies knowledge rules to incoming or existing information to create expert advice for the user.	Col. 4, lines 39-62: the master computer generates a thank you letter to the customer visited using the logo, return address and letter content of the respective employer.

U.S. Pat. No. 6,067,525 Claim 1	U.S. Pat. No. 5,450,314	U.S. Pat. No. 4,567,359
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see FIG. 2 and col. 6, lines 59-61: A data processing method and system, which can be used in department or clothing stores as a sales promotion tool for face to face sales.	The preamble is not a limitation. Nonetheless, see Abstract; col. 3, lines 49-56; col. 9, lines 13-20 & 31-35: a computerized system applied to many types of customer service and sales industries.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Col. 7, lines 1-5: FIG. 5 shows a data accumulation subsystem and FIG. 6 shows a data use subsystem to facilitate the system to propose recommendations of coordinates for women's clothes in a face to face sale (col. 10, lines 40-44 and lines 58-65).	Col. 5, lines 37-55; col. 3, lines 5-8: a central data processing center tied to remote insurance company terminals, transaction terminals, motor vehicle service bureaus and credit information and bank terminals for the preparation, verification and forwarding of insurance quotations and policy execution.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	FIG. 1; col. 4, lines 21-60, and col. 9, lines 28-32: Processing Unit 201, and subunits 101-106, accumulate kansai information, create and display a proposed object therefrom, and generate alternative adjectives based upon buyer interaction. Col. 10, lines 1-23: proposed wardrobe coordinates are displayed (event) and user denotes satisfaction/no satisfaction for the elements (change in state of event) w/i the use subsystem of FIG. 6	Col. 7, line 61 to col. 8, line 2: central processor (event manager) detects changes in state of event (request for quotation).
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Col. 10, lines 1-23: context of event is user's previously accumulated kansai, which is analyzed as a result of the detected changes in state (user's satisfaction/no satisfaction selection) of the event (proposed wardrobe coordinates).	Col. 7, line 61 to col. 8, line 2: context of event (request for quotation) is the type of quote requested.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	Col. 10, lines 1-23: the operation of analyzing the user profile/attribute info in the kansai database 1201 of the accumulation subsystem is auto initiated to propose substitutes (new action) for the elements deemed "no satisfaction" based upon user kansai suggesting high likelihood that proposed substitute would be accepted as adaptable with "satisfaction" elements.	Col. 7, line 61 to col. 8, line 2: auto initiation of location of applicable ratings info from insurance company terminal subsystems to facilitate action of making insurance calculation and providing quotation for each insurance company based on type of quote requested and "based on the information received from the customer." (col. 7, line 68).

U.S. Pat. No. 6,067,525 Claim 1	U.S. Pat. No. 5,216,592	U.S. Pat. No. 5,283,856
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Col. 2, lines 2-9 & 37-38: data processing system for tracking items through a business process; for creating state based automation environments; and to process order items (thereby facilitating a sales process).	The preamble is not a limitation. Nonetheless, see 2:40-43 Present invention provides a flexible, efficient, event-driven and conditional rule-based system which can be transparently implemented for use, e.g., in electronic mail applications; Messaging Systems are sales systems used to facilitate a sales process.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Col. 2, lines 51-61: definition facility kernel 104 & execution facility kernel 110.	6:24-47 Events can be synthesized based on inter-application communication (IAC). Microsoft DDE permits the exchange of data between disparate applications programs. Other applications can exchange data or parameters with a DDE interface, all performed during a phase of a sales process.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	Col. 3, lines 64-68 – the process flow controller. See col. 4, lines 11-25: process flow controller controls task assignment and activation based upon item state and associated data. See col. 5, lines 1-20: after activity 206 completes ... Completion is the state characteristic of the activity, which is the event.	6:24-47 An event synthesizer component of the event manager, responsive to the DDE, creates an IAC event. The IAC event synthesizer uses an event record which includes client information. 8:3-5 Event manager 24 interfaces with the rest of the system. 9:27-30 When the event occurs, the event manager 24 fetches the event.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Col. 5, lines 1-20: Collector asks for analysis (status update) concerning completed activity. State change process flow 162 controls situations where multiple parallel activities are required to be completed before processing of a subsequent step.	8:38-41 the occurrence of events causes selected conditions to be tested; when satisfied, the conditions cause actions to be taken; and actions in turn may lead to the occurrence of new events. 4:36-38 Various events can be specified to trigger evaluation of a condition and invocation of corresponding actions. 9:27-44 The rule engine 38, given the event occurrence and having the rule and associated message, then executes the rule by effecting performance of the specified action.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	Col. 5, lines 1-20: In analyze value step 212, a determination may be made that the requirement is not cost justified, leading to a rejection step 216, ..., if step 212 leads to a rejection, then activity 214 will never be performed.	

U.S. Pat. No. 6,067,525 Claim 1	GoldMine Software v. 2.5a & Reference Manual Release 1994 – Manual ©1996	GoldMine Factsback v.2.5a Bulletin #371 – Nov. 15, 1993
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Chap. 1, generally, and p.1: GoldMine maintains a database of information on contacts, prospective clients and current customers...A complete history of your interactions...holds detailed information on each contact such as prior sales, telephone calls or prior shipment...you can effectively track all of your commitments...Additionally, when Goldmine is used by multiple individuals on a network or notebooks...	The preamble is not a limitation. Nonetheless, see p.1: Automated Processes (AP) is a system to program GoldMine to perform actions automatically; e.g., schedule activities, add history, print mailings, and fax info...in the hope of making a sale
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Chap. 1, generally, and pp.1-3: GoldMine automates the following key areas of daily business activity: client/contact management (client contact and profile databases); time management (calendar/day timer; task management (via users on network or remote notebook computers); document management (document management and data exchange programs); and sales staff management (sales staff statistical databases).	p.1: tracks are made up of events. An event is a numbered line in your track that does something when a condition is met. Sequential events perform different actions in sequence; e.g, print a letter, schedule a call back.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	Chap. 16, generally, and p. 227: GoldMine provides functionality that allows you to process tracks for all or a set of contact records. GoldMine scans the database for contact records having attached tracks. When found, GoldMine first determines if a track contains a preemptive event. Next, GoldMine evaluates trigger condition of the current event to determine if the event should be processed. If triggered, the event's action is performed, and the next event is evaluated. When the track has no longer any outstanding events, GoldMine looks at the next track, or the next contact record.	What an event does is called an action. The condition which controls whether the action is taken is called a trigger. The trigger for a sequential event must be met, and its action taken, before GoldMine will evaluate the next sequential event in the track.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Chap. 16, generally, and p. 222: an Automated Process (AP) is referred to as a track. Tracks consist of ...two or more events, which are step by step instructions GoldMine must evaluate to perform a selected series of activities. Each event is composed of an action and a trigger. Actions, such as printing a letter or scheduling an activity, are performed based on pre-defined conditions, which are triggers that cause the actions to be executed. p. 223: Consider event – wait 10 days, print letter. The trigger is the passage of time. To determine when the action should be executed, GoldMine will start counting days after the completion of the prior event. p. 233-238: Each AP event performs a specific action based on a trigger. GoldMine evaluates triggers of the current event...when triggered, the event's action is performed and processing continues to the next event. The following trigger conditions can be used: elapsed days, immediate, profile record, history record, scheduled activity and Xbase condition. Profile, history,	p.1: Preemptive Events are usually conditions for ending a track...if you have a ...series of sequential events...callbacks sending prospective letters in hope of making a sale, and the sale is made; you should use a preemptive event to remove the track and stop GoldMine from sending the prospecting letters to that contact.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.		p.2: Planning Tracks of AP: determine what conditions could change under which you would want the track to end, for example, if the contact orders the products that your track is promoting,

	<p>scheduled activity and xBase condition can be preemptive events, specifying that the processing of the event's action cannot occur until a specified profile record, history record, other scheduled activity or a xBase expression occurs or is evaluated as true.</p> <p>Chap. 16, generally, and p. 221: The real power of Automated Processes ...automatically perform ...repetitive tasks required to close medium to long-length sales cycles.</p>	<p>you might want to add another track to sell other products to the contact and of course you should remove the original track so that the contact does not receive further promotion on a product already bought.</p>
--	---	---

U.S. Pat. No. 6,067,525 Claim 1	Casanova, et al., “Expert System for Automatic Authorization of Deficits.” IEEE, 1989.	The Johnson Software, 1983
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract & p.732:col.1:lls1-5: Expert System for financial companies aiming for automatic authorization of debit operations when customer has insufficient funds is a sales system used to facilitate the sales process of selling a loan	The preamble is not a limitation. Nonetheless, see Transcript of June 9, 2008 Deposition of Jerome Johnson (“Johnson Tr. 1”) at 47:10-21, 69:1-70:14 The Johnson Software is a sales system, implemented by computer and used to facilitate sales. Transcript of June 10, 2008 Deposition of Jerome Johnson (“Johnson Tr. 2”) at 227:3-11.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	p.733:col.2:line 38 thru p.735:col.2:line60 & Fig. 2: A stratified knowledge base exists, where each stratum is formed of rule bases capable of decision making using information known therein. Discriminatory and accumulative knowledges exist. Discriminatory knowledge refers to info permitting immediate decision-making (e.g., various account balances and activity). Accumulative knowledge refers to various client characteristics, where each accumulative knowledge subsystem can evaluate such factors as client delinquency, economic solvency, customer profile, payment capacity, client condition, client image, etc. All factor evaluations facilitate decisions made during the process of selling a loan.	Johnson Tr. 2 at 227:3-11. The Johnson Software has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. For example, the “pricing” module facilitates the action of determining the price for the product requested by the customer during the pricing phase of the sales process. Johnson Tr. 1 at 100:5-15; Johnson Tr. 2 at 227:3-11; Johnson Tr. 228:10-22.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	p.733:col.2:lls26-31 & Fig.1: An expert system is used, including inference engine, data base, logic control and graphic representation. An explication module of the expert system interacts with the stratified knowledge database and subsystems to factor contexts associated with the event to make a decision on the sale of the loan. p.733:col.2:lls20-22: the entry from a client into the system of a petition for overdraft protection, including the requested amount and date of request, are changes in state characteristic of an event occurring within the system.	The Johnson Software discussed in the Johnson deposition has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. In addition, Mr. Johnson testified that the Johnson Software includes features and/or functionality that are the same as the functionality of the event manager, for example the Johnson Software transfers information from one module to another, thereby illustrating the Johnson Software’s ability to detect a change in state in one module and thereafter incorporate that change into subsequent modules. Tr. 1 at 101:2-10; Johnson Tr. 2 at 227:3-232:25. Moreover, the language “detecting one or more changes in state characteristic of an event” means the same as

		“recognizing an event.” Johnson Tr. 2 at 212:21-213:6.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Upon occurrence of the event (the petition for overdraft protection, at p.733:col.2:lls20-22), see p.733:col.1:lls45-52: a subjective study of the account is performed, attempting to identify the customer, his financial means, his habits, his personal expenses,...., and other factors constituting an economic and psychological profile. All are contexts in which the event occurred.	<p>According to Mr. Johnson, “inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state” means the same as “determining a context in which the recognized event occurs.” Johnson Tr. 2 at 214:18-215:20.</p> <p>The Johnson Software infers the occurrence of an event, such as, for example, by inferring a customer contact (the “event”) and the context of the event by detecting when data has been entered into one “module” (“detected changes in state”). Johnson Tr. 1 at 101:2-10; Johnson Tr. 2 at 201:2-16; 232:16-25.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>p.734:col.1:lls30-33: From each stratum of the accumulative knowledge, some quality of the client is obtained in response to occurrence of the event and contexts associated therewith, along with the ultimate decision on the sale.</p> <p>p. 736:col.1:lls44-54: The system is automatic, so there is no need for any human intervention in the decision-making process. All information necessary is taken from the Entity’s information services network. The system performs a personalized treatment of the operations between the client and the bank based on heuristics, just as an expert human would.</p>	The Johnson Software discussed in the Johnson deposition has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. One noted feature of the Johnson Software is that data that was entered into one module was <i>automatically</i> transferred to other modules within the system and could be used by those modules to assist the salesperson in the sales process. (Johnson Tr. 2 at 232:16-25)

U.S. Pat. No. 6,067,525 Claim 20	Mott's Business-Management Expert Planning System	U.S. Pat. No. 4,853,852
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see §2.4 is a Sales-Programme Subsystem within an EPS and business management planning system (See Abstract). See ¶37: A tool to predict which prospects will materialize and how soon.</p> <p>Abstract & ¶3: four subsystems are resource scheduling, market-intelligence, technical expertise and sale programme. See ¶6: expert system includes knowledge database, an inference engine and information database. See ¶11 & FIG. 5: The Information database includes the four subsystems as well.</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, FIG. 1 and col. 3, lines 4-11: A marketing tool having a portable lap computer for recording a sales call, and a master computer to automatically prepare planners</p> <p>FIG. 1 and col. 3, lines 19-31: lap computer of each salesperson, each employer computer, the master computer system and all related databases are the subsystems.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	§ 4: Inference Engine is the event manager, coupled to knowledge and information databases with four subsystems.	FIG. 1 & col. 3, lines 19-24 – the master computer facility is the event manager, which detects the day's customer call information from each salesperson of each employer (each visit is an event).
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>¶39 & FIG. 10: Table of Prospect Attributes - ¶25: Market Intelligence. ¶27: In obtaining and analyzing market intelligence, trends about changing markets are identified, providing pointers to changes in technical-expertise subsystem.</p> <p>¶40 & FIG. 11: Model's product range is event occurrence, context is prospect table of attributes.</p>	Col. 4, lines 39-62: the master computer infers event occurrence (customer visit) and a context of the event (the employer of each salesperson).
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>¶40 & FIG. 11: A more comprehensive product package is developed from event occurrence and attribute information.</p> <p>¶44: Inference engine uses and applies knowledge rules to incoming or existing information to create expert advice for the user.</p>	Col. 4, lines 39-62: the master computer generates a thank you letter to the customer visited using the logo, return address and letter content of the respective employer.

U.S. Pat. No. 6,067,525 Claim 20	U.S. Pat. No. 5,450,314	U.S. Pat. No. 4,567,359
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see FIG. 2 and col. 6, lines 59-61: A data processing method and system, which can be used in department or clothing stores as a sales promotion tool for face to face sales.</p> <p>Col. 7, lines 1-5: FIG. 5 shows a data accumulation subsystem and FIG. 6 shows a data use subsystem to facilitate the system to propose recommendations of coordinates for women's clothes in a face to face sale (col. 10, lines 40-44 and lines 58-65).</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract; col. 3, lines 49-56; col. 9, lines 13-20 & 31-35: a computerized system applied to many types of customer service and sales industries.</p> <p>Col. 5, lines 37-55; col. 3, lines 5-8: a central data processing center tied to remote insurance company terminals, transaction terminals, motor vehicle service bureaus and credit information and bank terminals for the preparation, verification and forwarding of insurance quotations and policy execution.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	<p>FIG. 1; col. 4, lines 21-60, and col. 9, lines 28-32: Processing Unit 201, and subunits 101-106, accumulate kansai information, create and display a proposed object therefrom, and generate alternative adjectives based upon buyer interaction.</p> <p>Col. 10, lines 1-23: proposed wardrobe coordinates are displayed (event) and user denotes satisfaction/no satisfaction for the elements (change in state of event) w/i the use subsystem of FIG. 6</p>	<p>Col. 7, line 61 to col. 8, line 2: central processor (event manager) detects changes in state of event (request for quotation).</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>Col. 10, lines 1-23: context of event is user's previously accumulated kansai, which is analyzed as a result of the detected changes in state (user's satisfaction/no satisfaction selection) of the event (proposed wardrobe coordinates).</p>	<p>Col. 7, line 61 to col. 8, line 2: context of event (request for quotation) is the type of quote requested.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>Col. 10, lines 1-23: the operation of analyzing the user profile/attribute info in the kansai database 1201 of the accumulation subsystem is auto initiated to propose substitutes (new action) for the elements deemed "no satisfaction" based upon user kansai suggesting high likelihood that proposed substitute would be accepted as adaptable with "satisfaction" elements.</p>	<p>Col. 7, line 61 to col. 8, line 2: auto initiation of location of applicable ratings info from insurance company terminal subsystems to facilitate action of making insurance calculation and providing quotation for each insurance company based on type of quote requested and "based on the information received from the customer." (col. 7, line 68).</p>

U.S. Pat. No. 6,067,525 Claim 20	U.S. Pat. No. 5,216,592	U.S. Pat. No. 5,283,856
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Col. 2, lines 2-9 & 37-38: data processing system for tracking items through a business process; for creating state based automation environments; and to process order items (thereby facilitating a sales process).</p> <p>Col. 2, lines 51-61: definition facility kernel 104 & execution facility kernel 110.</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see 2:40-43 Present invention provides a flexible, efficient, event-driven and conditional rule-based system which can be transparently implemented for use, e.g., in electronic mail applications; Messaging Systems are sales systems used to facilitate a sales process.</p> <p>6:24-47 Events can be synthesized based on inter-application communication (IAC). Microsoft DDE permits the exchange of data between disparate applications programs. Other applications can exchange data or parameters with a DDE interface, all performed during a phase of a sales process.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	Col. 3, lines 64-68 – the process flow controller. See col. 4, lines 11-25: process flow controller controls task assignment and activation based upon item state and associated data. See col. 5, lines 1-20: after activity 206 completes ... Completion is the state characteristic of the activity, which is the event.	6:24-47 An event synthesizer component of the event manager, responsive to the DDE, creates an IAC event. The IAC event synthesizer uses an event record which includes client information. 8:3-5 Event manager 24 interfaces with the rest of the system. 9:27-30 When the event occurs, the event manager 24 fetches the event.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Col. 5, lines 1-20: Collector asks for analysis (status update) concerning completed activity. State change process flow 162 controls situations where multiple parallel activities are required to be completed before processing of a subsequent step.	8:38-41 the occurrence of events causes selected conditions to be tested; when satisfied, the conditions cause actions to be taken; and actions in turn may lead to the occurrence of new events. 4:36-38 Various events can be specified to trigger evaluation of a condition and invocation of corresponding actions. 9:27-44 The rule engine 38, given the event occurrence and having the rule and associated message, then executes the rule by effecting performance of the specified action.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	Col. 5, lines 1-20: In analyze value step 212, a determination may be made that the requirement is not cost justified, leading to a rejection step 216, ..., if step 212 leads to a rejection, then activity 214 will never be performed.	

U.S. Pat. No. 6,067,525 Claim 20	GoldMine Software v. 2.5a & Reference Manual Release 1994 – Manual ©1996	GoldMine Factsback v.2.5a Bulletin #371 – Nov. 15, 1993
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Chap. 1, generally, and p.1: GoldMine maintains a database of information on contacts, prospective clients and current customers...A complete history of your interactions...holds detailed information on each contact such as prior sales, telephone calls or prior shipment...you can effectively track all of your commitments...Additionally, when Goldmine is used by multiple individuals on a network or notebooks...</p> <p>Chap. 1, generally, and pp.1-3: GoldMine automates the following key areas of daily business activity: client/contact management (client contact and profile databases); time management (calendar/day timer; task management (via users on network or remote notebook computers); document management (document management and data exchange programs); and sales staff management (sales staff statistical databases).</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see p.1: Automated Processes (AP) is a system to program GoldMine to perform actions automatically; e.g., schedule activities, add history, print mailings, and fax info...in the hope of making a sale</p> <p>p.1: tracks are made up of events. An event is a numbered line in your track that does something when a condition is met. Sequential events perform different actions in sequence; e.g, print a letter, schedule a call back.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	Chap. 16, generally, and p. 227: GoldMine provides functionality that allows you to process tracks for all or a set of contact records. GoldMine scans the database for contact records having attached tracks. When found, GoldMine first determines if a track contains a preemptive event. Next, GoldMine evaluates trigger condition of the current event to determine if the event should be processed. If triggered, the event's action is performed, and the next event is evaluated. When the track has no longer any outstanding events, GoldMine looks at the next track, or the next contact record.	What an event does is called an action. The condition which controls whether the action is taken is called a trigger. The trigger for a sequential event must be met, and its action taken, before GoldMine will evaluate the next sequential event in the track.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Chap. 16, generally, and p. 222: an Automated Process (AP) is referred to as a track. Tracks consist of ...two or more events, which are step by step instructions GoldMine must evaluate to perform a selected series of activities. Each event is composed of an action and a trigger. Actions, such as printing a letter or scheduling an activity, are performed based on pre-defined conditions, which are triggers that cause the actions to be executed. p. 223: Consider event – wait 10 days, print letter. The trigger is the passage of time. To determine when the action should be executed, GoldMine will start counting days after the completion of the prior event. p. 233-238: Each AP event performs a specific action based on a trigger. GoldMine evaluates triggers of the current event...when triggered, the event's action is performed and processing continues to the next event. The following trigger conditions can be used: elapsed days, immediate, profile record, history record, scheduled activity and Xbase condition. Profile, history,	p.1: Preemptive Events are usually conditions for ending a track...if you have a ...series of sequential events...callbacks sending prospective letters in hope of making a sale, and the sale is made; you should use a preemptive event to remove the track and stop GoldMine from sending the prospecting letters to that contact.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.		p.2: Planning Tracks of AP: determine what conditions could change under which you would want the track to end, for example, if the contact orders the products that your track is promoting,

	<p>scheduled activity and xBase condition can be preemptive events, specifying that the processing of the event's action cannot occur until a specified profile record, history record, other scheduled activity or a xBase expression occurs or is evaluated as true.</p> <p>Chap. 16, generally, and p. 221: The real power of Automated Processes ...automatically perform ...repetitive tasks required to close medium to long-length sales cycles.</p>	<p>you might want to add another track to sell other products to the contact and of course you should remove the original track so that the contact does not receive further promotion on a product already bought.</p>
--	---	---

U.S. Pat. No. 6,067,525 Claim 20	Casanova, et al., "Expert System for Automatic Authorization of Deficits." IEEE, 1989.
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract & p.732:col.1:lls1-5: Expert System for financial companies aiming for automatic authorization of debit operations when customer has insufficient funds is a sales system used to facilitate the sales process of selling a loan</p> <p>p.733:col.2:line 38 thru p.735:col.2:line60 & Fig. 2: A stratified knowledge base exists, where each stratum is formed of rule bases capable of decision making using information known therein. Discriminatory and accumulative knowledges exist. Discriminatory knowledge refers to info permitting immediate decision-making (e.g., various account balances and activity). Accumulative knowledge refers to various client characteristics, where each accumulative knowledge subsystem can evaluate such factors as client delinquency, economic solvency, customer profile, payment capacity, client condition, client image, etc. All factor evaluations facilitate decisions made during the process of selling a loan.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	<p>p.733:col.2:lls26-31 & Fig.1: An expert system is used, including inference engine, data base, logic control and graphic representation. An explication module of the expert system interacts with the stratified knowledge database and subsystems to factor contexts associated with the event to make a decision on the sale of the loan.</p> <p>p.733:col.2:lls20-22: the entry from a client into the system of a petition for overdraft protection, including the requested amount and date of request, are changes in state characteristic of an event occurring within the system.</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>Upon occurrence of the event (the petition for overdraft protection, at p.733:col.2:lls20-22), see p.733:col.1:lls45-52: a subjective study of the account is performed, attempting to identify the customer, his financial means, his habits, his personal expenses,..., and other factors constituting an economic and psychological profile. All are contexts in which the event occurred.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>p.734:col.1:lls30-33: From each stratum of the accumulative knowledge, some quality of the client is obtained in response to occurrence of the event and contexts associated therewith, along with the ultimate decision on the sale.</p> <p>p. 736:col.1:lls44-54: The system is automatic, so there is no need for any human intervention in the decision-making process. All information necessary is taken from the Entity's information services network. The system performs a personalized treatment of the operations between the client and the bank based on heuristics, just as an expert human would.</p>

U.S. Pat. No. 6,067,525 Claim 20	Jerome Johnson Unnamed Software, 1983
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Transcript of June 9, 2008 Deposition of Jerome Johnson (“Johnson Tr. 1”) at 47:10-21, 69:1-70:14 The Johnson Software is a sales system, implemented by computer and used to facilitate sales. Johnson Tr. 2 at 227:3-11.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	<p>The language “detecting one or more changes in state characteristic of an event” means the same as “recognizing an event.” Johnson Tr. 2 at 212:21-213:6.</p> <p>The Johnson Software discussed in the Johnson deposition has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. The Johnson Software detects one or more changes in state characteristic of an event and/or recognizes an event by, for example, automatically detecting that the sales process has transitioned from the “pricing” module to the “quoting” module and transferring information from one module to another, thereby illustrating the Johnson Software’s ability to detect a change in state in one module and thereafter incorporate that change into subsequent modules. Tr. 1 at 101:2-10; Johnson Tr. 2 at 227:3-232:25.</p> <p>The Johnson Software could detect when information was input into one module and then transfer that information to another module, thereby illustrating the Johnson Software’s ability to detect a change in state in one module and thereafter automatically incorporate that change into subsequent modules. In other words, no human intervention was required. (Johnson Tr. 2 at 232:16-25).</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>According to Mr. Johnson, “inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state” means the same as “determining a context in which the recognized event occurs.” Johnson Tr. 2 at 214:18-215:20.</p> <p>The Johnson Software infers the occurrence of an event, such as, for example, by inferring a customer contact (the “event”) and the context of the event by detecting when data has been entered into one “module” (“detected changes in state”). Johnson Tr. 1 at 101:2-10; Johnson Tr. 2 at 201:2-16; 232:16-25.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>The Johnson Software discussed in the Johnson deposition has multiple “modules”, each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. One noted feature of the Johnson Software is that data that was entered into one module was automatically transferred to other modules within the system and could be used by those modules to assist the salesperson in the sales process. (Johnson Tr. 2 at 232:16-25)</p>

U.S. Pat. No. 6,067,525 Claim 40	Mott's Business-Management Expert Planning System	U.S. Pat. No. 4,853,852
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see §2.4 is a Sales-Programme Subsystem within an EPS and business management planning system (See Abstract). See ¶37: A tool to predict which prospects will materialize and how soon	The preamble is not a limitation. Nonetheless, see Abstract, FIG. 1 and col. 3, lines 4-11: A marketing tool having a portable lap computer for recording a sales call, and a master computer to automatically prepare planners
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Abstract & ¶3: four subsystems are resource scheduling, market-intelligence, technical expertise and sale programme. See ¶6: expert system includes knowledge database, an inference engine and information database. See ¶11 & FIG. 5: The Information database includes the four subsystems as well.	FIG. 1 and col. 3, lines 19-31: lap computer of each salesperson, each employer computer, the master computer system and all related databases are the subsystems.
an event manager, coupled to the subsystems and configured to	§ 4: Inference Engine is the event manager, coupled to knowledge and information databases with four subsystems.	FIG. 1 & col. 3, lines 19-24 – the master computer facility is the event manager, which detects the day's customer call information from each salesperson of each employer (each visit is an event).
detect one or more changes in state characteristic of an event occurring within the system,		
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	¶39 & FIG. 10: Table of Prospect Attributes - ¶25: Market Intelligence. ¶27: In obtaining and analyzing market intelligence, trends about changing markets are identified, providing pointers to changes in technical-expertise subsystem. ¶40 & FIG. 11: Model's product range is event occurrence, context is prospect table of attributes.	Col. 4, lines 39-62: the master computer infers event occurrence (customer visit) and a context of the event (the employer of each salesperson).
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and		
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	¶40 & FIG. 11: A more comprehensive product package is developed from event occurrence and attribute information. ¶44: Inference engine uses and applies knowledge rules to incoming or existing information to create expert advice for the user.	Col. 4, lines 39-62: the master computer generates a thank you letter to the customer visited using the logo, return address and letter content of the respective employer.

U.S. Pat. No. 6,067,525 Claim 40	U.S. Pat. No. 5,450,314	U.S. Pat. No. 4,567,359
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see FIG. 2 and col. 6, lines 59-61: A data processing method and system, which can be used in department or clothing stores as a sales promotion tool for face to face sales.	The preamble is not a limitation. Nonetheless, see Abstract; col. 3, lines 49-56; col. 9, lines 13-20 & 31-35: a computerized system applied to many types of customer service and sales industries.
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Col. 7, lines 1-5: FIG. 5 shows a data accumulation subsystem and FIG. 6 shows a data use subsystem to facilitate the system to propose recommendations of coordinates for women's clothes in a face to face sale (col. 10, lines 40-44 and lines 58-65).	Col. 5, lines 37-55; col. 3, lines 5-8: a central data processing center tied to remote insurance company terminals, transaction terminals, motor vehicle service bureaus and credit information and bank terminals for the preparation, verification and forwarding of insurance quotations and policy execution.
an event manager, coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring within the system,	FIG. 1; col. 4, lines 21-60, and col. 9, lines 28-32: Processing Unit 201, and subunits 101-106, accumulate kansai information, create and display a proposed object therefrom, and generate alternative adjectives based upon buyer interaction. Col. 10, lines 1-23: proposed wardrobe coordinates are displayed (event) and user denotes satisfaction/no satisfaction for the elements (change in state of event) w/i the use subsystem of FIG. 6	Col. 7, line 61 to col. 8, line 2: central processor (event manager) detects changes in state of event (request for quotation).
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Col. 10, lines 1-23: context of event is user's previously accumulated kansai, which is analyzed as a result of the detected changes in state (user's satisfaction/no satisfaction selection) of the event (proposed wardrobe coordinates).	Col. 7, line 61 to col. 8, line 2: context of event (request for quotation) is the type of quote requested.
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and		
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	Col. 10, lines 1-23: the operation of analyzing the user profile/attribute info in the kansai database 1201 of the accumulation subsystem is auto initiated to propose substitutes (new action) for the elements deemed "no satisfaction" based upon user kansai suggesting high likelihood that proposed substitute would be accepted as adaptable with "satisfaction" elements.	Col. 7, line 61 to col. 8, line 2: auto initiation of location of applicable ratings info from insurance company terminal subsystems to facilitate action of making insurance calculation and providing quotation for each insurance company based on type of quote requested and "based on the information received from the customer." (col. 7, line 68).

U.S. Pat. No. 6,067,525 Claim 40	U.S. Pat. No. 5,216,592	U.S. Pat. No. 5,283,856
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Col. 2, lines 2-9 & 37-38: data processing system for tracking items through a business process; for creating state based automation environments; and to process order items (thereby facilitating a sales process).</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see 2:40-43 Present invention provides a flexible, efficient, event-driven and conditional rule-based system which can be transparently implemented for use, e.g., in electronic mail applications; Messaging Systems are sales systems used to facilitate a sales process.</p>
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Col. 2, lines 51-61: definition facility kernel 104 & execution facility kernel 110.	6:24-47 Events can be synthesized based on inter-application communication (IAC). Microsoft DDE permits the exchange of data between disparate applications programs. Other applications can exchange data or parameters with a DDE interface, all performed during a phase of a sales process.
<p>an event manager, coupled to the subsystems and configured to</p> <p>detect one or more changes in state characteristic of an event occurring within the system,</p>	Col. 3, lines 64-68 – the process flow controller. See col. 4, lines 11-25: process flow controller controls task assignment and activation based upon item state and associated data. See col. 5, lines 1-20: after activity 206 completes ... Completion is the state characteristic of the activity, which is the event.	6:24-47 An event synthesizer component of the event manager, responsive to the DDE, creates an IAC event. The IAC event synthesizer uses an event record which includes client information. 8:3-5 Event manager 24 interfaces with the rest of the system. 9:27-30 When the event occurs, the event manager 24 fetches the event.
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and		
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and		

<p>automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.</p>	<p>Col. 5, lines 1-20: In analyze value step 212, a determination may be made that the requirement is not cost justified, leading to a rejection step 216, ..., if step 212 leads to a rejection, then activity 214 will never be performed.</p>	
---	--	--

U.S. Pat. No. 6,067,525 Claim 40	GoldMine Software v. 2.5a & Reference Manual Release 1994 – Manual ©1996	GoldMine Factsback v.2.5a Bulletin #371 – Nov. 15, 1993
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Chap. 1, generally, and p.1: GoldMine maintains a database of information on contacts, prospective clients and current customers...A complete history of your interactions...holds detailed information on each contact such as prior sales, telephone calls or prior shipment...you can effectively track all of your commitments...Additionally, when Goldmine is used by multiple individuals on a network or notebooks...	The preamble is not a limitation. Nonetheless, see p.1: Automated Processes (AP) is a system to program GoldMine to perform actions automatically; e.g., schedule activities, add history, print mailings, and fax info...in the hope of making a sale
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Chap. 1, generally, and pp.1-3: GoldMine automates the following key areas of daily business activity: client/contact management (client contact and profile databases); time management (calendar/day timer; task management (via users on network or remote notebook computers); document management (document management and data exchange programs); and sales staff management (sales staff statistical databases).	p.1: tracks are made up of events. An event is a numbered line in your track that does something when a condition is met. Sequential events perform different actions in sequence; e.g, print a letter, schedule a call back.
an event manager, coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring within the system,	Chap. 16, generally, and p. 227: GoldMine provides functionality that allows you to process tracks for all or a set of contact records. GoldMine scans the database for contact records having attached tracks. When found, GoldMine first determines if a track contains a preemptive event. Next, GoldMine evaluates trigger condition of the current event to determine if the event should be processed. If triggered, the event's action is performed, and the next event is evaluated. When the track has no longer any outstanding events, GoldMine looks at the next track, or the next contact record.	What an event does is called an action. The condition which controls whether the action is taken is called a trigger. The trigger for a sequential event must be met, and its action taken, before GoldMine will evaluate the next sequential event in the track.
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Chap. 16, generally, and p. 222: an Automated Process (AP) is referred to as a track. Tracks consist of ...two or more events, which are step by step instructions GoldMine must evaluate to perform a selected series of activities. Each event is composed of an action and a trigger. Actions, such as printing a letter or scheduling an activity, are performed based on pre-defined conditions, which are triggers that cause the actions to be executed. p. 223: Consider event – wait 10 days, print letter. The trigger is the passage of time. To determine when the action should be executed, GoldMine will start counting days after the completion of the prior event. p. 233-238: Each AP event performs a specific action based on a trigger. GoldMine evaluates triggers of the current event...when triggered, the event's action is performed and processing continues to the next event. The following trigger conditions can be used: elapsed days, immediate, profile record, history record, scheduled activity and Xbase condition. Profile, history,	p.1: Preemptive Events are usually conditions for ending a track...if you have a ...series of sequential events...callbacks sending prospective letters in hope of making a sale, and the sale is made; you should use a preemptive event to remove the track and stop GoldMine from sending the prospecting letters to that contact.
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and		
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.		p.2: Planning Tracks of AP: determine what conditions could change under which you would want the track to end, for example, if the contact orders the products that your track is promoting,

	<p>scheduled activity and xBase condition can be preemptive events, specifying that the processing of the event's action cannot occur until a specified profile record, history record, other scheduled activity or a xBase expression occurs or is evaluated as true.</p> <p>Chap. 16, generally, and p. 221: The real power of Automated Processes ...automatically perform ...repetitive tasks required to close medium to long-length sales cycles.</p>	<p>you might want to add another track to sell other products to the contact and of course you should remove the original track so that the contact does not receive further promotion on a product already bought.</p>
--	---	---

U.S. Pat. No. 6,067,525 Claim 40	Casanova, et al., “Expert System for Automatic Authorization of Deficits.” IEEE, 1989.
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract & p.732:col.1:lls1-5: Expert System for financial companies aiming for automatic authorization of debit operations when customer has insufficient funds is a sales system used to facilitate the sales process of selling a loan
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	p.733:col.2:line 38 thru p.735:col.2:line60 & Fig. 2: A stratified knowledge base exists, where each stratum is formed of rule bases capable of decision making using information known therein. Discriminatory and accumulative knowledges exist. Discriminatory knowledge refers to info permitting immediate decision-making (e.g., various account balances and activity). Accumulative knowledge refers to various client characteristics, where each accumulative knowledge subsystem can evaluate such factors as client delinquency, economic solvency, customer profile, payment capacity, client condition, client image, etc. All factor evaluations facilitate decisions made during the process of selling a loan.
an event manager, coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring within the system,	p.733:col.2:lls26-31 & Fig.1: An expert system is used, including inference engine, data base, logic control and graphic representation. An explication module of the expert system interacts with the stratified knowledge database and subsystems to factor contexts associated with the event to make a decision on the sale of the loan. p.733:col.2:lls20-22: the entry from a client into the system of a petition for overdraft protection, including the requested amount and date of request, are changes in state characteristic of an event occurring within the system.
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	Upon occurrence of the event (the petition for overdraft protection, at p.733:col.2:lls20-22), see p.733:col.1:lls45-52: a subjective study of the account is performed, attempting to identify the customer, his financial means, his habits, his personal expenses,..., and other factors constituting an economic and psychological profile. All are contexts in which the event occurred.
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	p.734:col.1:lls30-33: From each stratum of the accumulative knowledge, some quality of the client is obtained in response to occurrence of the event and contexts associated therewith, along with the ultimate decision on the sale. p. 736:col.1:lls44-54: The system is automatic, so there is no need for any human intervention in the decision-making process. All information necessary is taken from the Entity’s information services network. The system performs a personalized treatment of the operations between the client and the bank based on heuristics, just as an expert human would.

U.S. Pat. No. 6,067,525 Claim 40	Jerome Johnson Unnamed Software, 1983
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Transcript of June 9, 2008 Deposition of Jerome Johnson (“Johnson Tr. 1”) at 47:10-21, 69:1-70:14 The Johnson Software is a sales system, implemented by computer and used to facilitate sales. Johnson Tr. 2 at 227:3-11.
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Johnson Tr. 2 at 227:3-11. The Johnson Software has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. For example, the “pricing” module facilitates the action of determining the price for the product requested by the customer during the pricing phase of the sales process. Johnson Tr. 1 at 100:5-15; Johnson Tr. 2 at 227:3-11; Johnson Tr. 228:10-22.
an event manager, coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring within the system,	The Johnson Software discussed in the Johnson deposition has multiple “modules,” each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. In addition, Mr. Johnson testified that the Johnson Software includes features and/or functionality that are the same as the functionality of the event manager, for example the Johnson Software transfers information from one module to another, thereby illustrating the Johnson Software’s ability to detect a change in state in one module and thereafter incorporate that change into subsequent modules. Tr. 1 at 101:2-10; Johnson Tr. 2 at 227:3-232:25. Moreover, the language “detecting one or more changes in state characteristic of an event” means the same as “recognizing an event.” Johnson Tr. 2 at 212:21-213:6.
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	According to Mr. Johnson, “inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state” means the same as “determining a context in which the recognized event occurs.” Johnson Tr. 2 at 214:18-215:20.
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	The Johnson Software infers the occurrence of an event, such as, for example, by inferring a customer contact (the “event”) and the context of the event by detecting when data has been entered into one “module” (“detected changes in state”). Johnson Tr. 1 at 101:2-10; Johnson Tr. 2 at 201:2-16; 232:16-25 The inferred event is linked to an action to be performed based on prior sales experience because the relationship between the inferred event and the action to be performed is controlled by the rules programmed into the system, which necessarily rely on prior sales experience. Johnson Tr. 2 at 165:7-22.
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	The Johnson Software discussed in the Johnson deposition has multiple “modules”, each of which are used during different phases of the sales process and are configured to assist the salesperson in completing the actions associated with that phase of the sales process. One noted feature of the Johnson Software is that data that was entered into one module was <i>automatically</i> transferred to other modules within the system and could be used by those modules to assist the salesperson in the sales process. (Johnson Tr. 2 at 232:16-25)

U.S. 6,067,525	Cragun – U.S. 5,774,868
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract. An automated sales promotion selection system uses neural networks to identify promising sales promotions based on recent customer purchases.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Customer Information Device (Fig.1, 14), Billing Terminal 16, and Sales Promotion Output Device 17 (Col. 3, line 66-col. 4, line 27).
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	Computer system 12, Neural network purchase adviser subsystem 24, demographic prediction subsystem 25; The customer information devices 14 and billing terminals 16 communicate with the computer system 12 using an I/O interface 34. (col. 4 lines 28-52). As items are purchased in a store, the neural network purchase adviser subsystem is invoked (col. 4, lines 55-57). Sales made via telephone orders and/or in the telemarketing context can be used with the system. (col. 18, lines 16-20). The demographic prediction subsystem 25 predicts the customer population that can be expected to be within the store at any one time based on a variety of factors. (Col. 16, lines 58-66).
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	The purchase adviser 24 neural network automatically collects purchase transaction data, segments the purchase items of a particular customer purchase transaction into predetermined purchase classes that define groups of items ordinarily purchased together, and identifies items that belong to a purchase class but were missing from the purchase transaction. (Fig. 10, col. 11, line 36-col. 12, line 13; col. 18, lines 21-27). The demographic prediction subsystem 25 processes the collected data to generate output comprising a predicted customer population inside the store at a given time. The demographic subsystem processes the predicted customer population with another neural network of the subsystem to generate output comprising predicted purchases. That is, a listing of items that the subsystem predicts would be purchased by a typical customer at the given time. (Col. 17, lines 44-60).
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	The system then selects a sales promotion to suggest the purchase of a missing item that likely will result in an additional sale. (col. 18, lines 27-29.) The missing items can then be suggested by a sales clerk for purchase or can be the subject of an automatically produced promotion, such as a coupon that can be redeemed for a discounted purchase price. (col. 2, lines 56-59). The demographic prediction subsystem 25 provides the predicted sales purchase data to the purchase advisor subsystem and its neural networks. The purchase advisor subsystem will segment the purchase items into purchase classes and generate selected sales promotions, such as purchase suggestions. The selected sales promotions can be used on the

	<p>general customer population or for direct mail campaigns and the like, rather than the use described previously of targeting particular customers making purchases.</p> <p>The output comprising the predicted customer population in the store and the output comprising the predicted purchase transactions can be used independently of any use in the purchase advisor subsystem.</p> <p>It might be useful to a store manager to have a sense of customers that can be expected in a store at anyone time, or to have an understanding of what products can reasonably be expected to be purchased at a given time of day. (Col. 17, line 60-col. 18, line 15).</p>
--	---

U.S. 6,067,525	Cragun – U.S. 5,774,868
<p>20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract. An automated sales promotion selection system uses neural networks to identify promising sales promotions based on recent customer purchases.</p> <p>Customer Information Device (Fig.1, 14), Billing Terminal 16, and Sales Promotion Output Device 17 (Col. 3, line 66-col. 4, line 27).</p>
<p>automatically detecting one or more changes in state characteristic of an event occurring in the sales process,</p>	<p>Computer system 12, Neural network purchase adviser subsystem 24, demographic prediction subsystem 25;</p> <p>The customer information devices 14 and billing terminals 16 communicate with the computer system 12 using an I/O interface 34. (col. 4 lines 28-52).</p> <p>As items are purchased in a store, the neural network purchase adviser subsystem is invoked (col. 4, lines 55-57).</p> <p>Sales made via telephone orders and/or in the telemarketing context can be used with the system. (col. 18, lines 16-20).</p> <p>The demographic prediction subsystem 25 predicts the customer population that can be expected to be within the store at any one time based on a variety of factors. (Col. 16, lines 58-66).</p>
<p>inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and</p>	<p>The purchase adviser 24 neural network automatically collects purchase transaction data, segments the purchase items of a particular customer purchase transaction into predetermined purchase classes that define groups of items ordinarily purchased together, and identifies items that belong to a purchase class but were missing from the purchase transaction. (Fig. 10, col. 11, line 36-col. 12, line 13; col. 18, lines 21-27).</p> <p>The demographic prediction subsystem 25 processes the collected data to generate output comprising a predicted customer population inside the store at a given time. The demographic subsystem processes the predicted customer population with another neural network of the subsystem to generate output comprising predicted purchases. That is, a listing of items that the subsystem predicts would be purchased by a typical customer at the given time. (Col. 17, lines 44-60).</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	<p>The system then selects a sales promotion to suggest the purchase of a missing item that likely will result in an additional sale. (col. 18, lines 27-29.)</p> <p>The missing items can then be suggested by a sales clerk for purchase or can be the subject of an automatically produced promotion, such as a coupon that can be redeemed for a discounted purchase price. (col. 2, lines 56-59).</p> <p>The demographic prediction subsystem 25 provides the predicted sales purchase data to the purchase advisor subsystem and its neural networks. The purchase advisor subsystem will segment the purchase items into purchase classes and generate selected sales promotions, such as purchase suggestions. The selected sales promotions can be used on the general customer population or for direct mail campaigns and the like,</p>

	<p>rather than the use described previously of targeting particular customers making purchases.</p> <p>The output comprising the predicted customer population in the store and the output comprising the predicted purchase transactions can be used independently of any use in the purchase advisor subsystem.</p> <p>It might be useful to a store manager to have a sense of customers that can be expected in a store at anyone time, or to have an understanding of what products can reasonably be expected to be purchased at a given time of day. (Col. 17, line 60-col. 18, line 15).</p>
--	--

U.S. 6,067,525	Cragun – U.S. 5,774,868
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract. An automated sales promotion selection system uses neural networks to identify promising sales promotions based on recent customer purchases.
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Customer Information Device (Fig.1, 14), Billing Terminal 16, and Sales Promotion Output Device 17 (Col. 3, line 66-col. 4, line 27).
an event manager, coupled to the subsystems and configured to	Computer system 12, Neural network purchase adviser subsystem 24, demographic prediction subsystem 25; The customer information devices 14 and billing terminals 16 communicate with the computer system 12 using an I/O interface 34. (col. 4 lines 28-52). Sales made via telephone orders and/or in the telemarketing context can be used with the system. (col. 18, lines 16-20). The demographic prediction subsystem 25 predicts the customer population that can be expected to be within the store at any one time based on a variety of factors. (Col. 16, lines 58-66).
detect one or more changes in state characteristic of an event occurring within the system,	As items are purchased in a store, the neural network purchase adviser subsystem is invoked (col. 4, lines 55-57). The demographic prediction subsystem 25 predicts the customer population that can be expected to be within the store at any one time based on a variety of factors. (Col. 16, lines 58-66).
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	The purchase adviser 24 neural network automatically collects purchase transaction data, segments the purchase items of a particular customer purchase transaction into predetermined purchase classes that define groups of items ordinarily purchased together, and identifies items that belong to a purchase class but were missing from the purchase transaction. (Fig. 10, col. 11, line 36-col. 12, line 13; col. 18, lines 21-27). The demographic prediction subsystem 25 processes the collected data to generate output comprising a predicted customer population inside the store at a given time. The demographic subsystem processes the predicted customer population with another neural network of the subsystem to generate output comprising predicted purchases. That is, a listing of items that the subsystem predicts would be purchased by a typical customer at the given time. (Col. 17, lines 44-60).
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	The system then selects a sales promotion to suggest the purchase of a missing item that likely will result in an additional sale. (col. 18, lines 27-29.) The missing items can then be suggested by a sales clerk for purchase or can be the subject of an automatically produced promotion, such as a coupon that can be redeemed for a discounted purchase price. (col. 2, lines 56-59). The demographic prediction subsystem 25 provides the predicted sales purchase data to the purchase advisor subsystem and its neural networks.

	<p>The purchase advisor subsystem will segment the purchase items into purchase classes and generate selected sales promotions, such as purchase suggestions. The selected sales promotions can be used on the general customer population or for direct mail campaigns and the like, rather than the use described previously of targeting particular customers making purchases.</p> <p>The output comprising the predicted customer population in the store and the output comprising the predicted purchase transactions can be used independently of any use in the purchase advisor subsystem.</p> <p>It might be useful to a store manager to have a sense of customers that can be expected in a store at anyone time, or to have an understanding of what products can reasonably be expected to be purchased at a given time of day. (Col. 17, line 60-col. 18, line 15).</p>
<p>automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.</p>	<p>The system then selects a sales promotion to suggest the purchase of a missing item that likely will result in an additional sale. (col. 18, lines 27-29.)</p> <p>The missing items can then be suggested by a sales clerk for purchase or can be the subject of an automatically produced promotion, such as a coupon that can be redeemed for a discounted purchase price. (col. 2, lines 56-59).</p> <p>The demographic prediction subsystem 25 provides the predicted sales purchase data to the purchase advisor subsystem and its neural networks. The purchase advisor subsystem will segment the purchase items into purchase classes and generate selected sales promotions, such as purchase suggestions. The selected sales promotions can be used on the general customer population or for direct mail campaigns and the like, rather than the use described previously of targeting particular customers making purchases.</p> <p>The output comprising the predicted customer population in the store and the output comprising the predicted purchase transactions can be used independently of any use in the purchase advisor subsystem.</p> <p>It might be useful to a store manager to have a sense of customers that can be expected in a store at anyone time, or to have an understanding of what products can reasonably be expected to be purchased at a given time of day. (Col. 17, line 60-col. 18, line 15).</p>

U.S. 6,067,525	Deaton – U.S. 5,201,010
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, col. 6, lines 4-24. A method and system is disclosed for performing targeted marketing on infrequent shoppers. The system collects and accumulates selected additional transactional data, thus allowing the store to adopt risk management approach to check verification tailored to the store's particular customer and financial situation, and allowing the store to develop customer profiles and to target advertising, marketing and promotions, and otherwise improve customer relations.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Check transaction Processing system (CTPS, col. 8, line 47- col. 20, line 31) Targeted Marketing col. (59 line 25 –col. 78, line 48)
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	CTPS contains multiple subsystems including an Event Manger (col. 50, line 32 – col. 56, line 19) Event driven activities (col. 31, line 51- col. 32, line 32).
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	1) Marketing based on range of last shopping dates (Col. 66, line 31 – col. 67, line 55. Fig. 16) Event – sale with a check / check processing Context – customer's last shopping date / customer's shopping history 2) Dissemination of Point of Sale coupons and direct mail coupons based upon shopping history (Col. 67, line 32 – col. 70, line 46. Fig. 17) Event – sale with a check / check processing Context – whether the user is a high volume shopper, primary shopper, or secondary (infrequent) shopper 3) Dissemination of point of sale coupons and direct mail coupons based upon scanned data (Col. 70, line 47 – col. 73, line 48. Fig. 18) Event – sale with a check / check processing Context – particular store departments in which the products being purchased
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	1) If this customer's last shopping date falls within the range of preselected shopping date range write this record to target file. Selection may also include a minimal dollar amount in a preselected time range. A promotion may then be selectively offered by the retail establishment to customers within the second database. (Col. 66, line 31 – col. 67, line 55. Fig. 16) 2) If secondary shopper dispense coupon value A (high incentive to make secondary shopper primary shopper) If primary shopper dispense coupon value B (lesser incentive package to maintain consistency) If high volume shopper dispense coupon value C (high value to hold on

	<p>to especially good shopper)</p> <p>Two method for determining coupon package: off-line (next visit) and on-line (time of sale). Coupon dispense through clerk or direct mail. (Col. 67, line 32 – col. 70, line 46. Fig. 17)</p> <p>3) system decide whether to award a coupon and what type of coupon to award.</p> <p>e.g. If shopper consistently does not buy from deli, generate coupon to induce customer to shop at the deli.</p> <p>Stored data (shopping history can also be used to generate targeted coupon.</p> <p>(Col. 70, line 47 – col. 73, line 48. Fig. 18)</p>
--	--

U.S. 6,067,525	Deaton – U.S. 5,201,010
<p>20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:</p>	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, col. 6, lines 4-24. A method and system is disclosed for performing targeted marketing on infrequent shoppers. The system collects and accumulates selected additional transactional data, thus allowing the store to adopt risk management approach to check verification tailored to the store's particular customer and financial situation, and allowing the store to develop customer profiles and to target advertising, marketing and promotions, and otherwise improve customer relations.</p> <p>Check transaction Processing system (CTPS, col. 8, line 47- col. 20, line 31) Targeted Marketing col. (59 line 25 –col. 78, line 48)</p>
<p>automatically detecting one or more changes in state characteristic of an event occurring in the sales process,</p>	<p>Event driven activities (col. 31, line 51- col. 32, line 32).</p>
<p>inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and</p>	<p>1) Marketing based on range of last shopping dates (Col. 66, line 31 – col. 67, line 55. Fig. 16) Event – sale with a check / check processing Context – customer's last shopping date / customer's shopping history</p> <p>2) Dissemination of Point of Sale coupons and direct mail coupons based upon shopping history (Col. 67, line 32 – col. 70, line 46. Fig. 17) Event – sale with a check / check processing Context – whether the customer is a high volume shopper, primary shopper, or secondary (infrequent) shopper</p> <p>3) Dissemination of point of sale coupons and direct mail coupons based upon scanned data (Col. 70, line 47 – col. 73, line 48. Fig. 18) Event – sale with a check / check processing Context – particular store departments in which the products being purchased</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	<p>1) If this customer's last shopping date falls within the range of preselected shopping date range write this record to target file. Selection may also include a minimal dollar amount in a preselected time range. A promotion may then be selectively offered by the retail establishment to customers within the second database. (Col. 66, line 31 – col. 67, line 55. Fig. 16)</p> <p>2) If secondary shopper dispense coupon value A (high incentive to make secondary shopper primary shopper) If primary shopper dispense coupon value B (lesser incentive package to maintain consistency) If high volume shopper dispense coupon value C (high value to hold on to especially good shopper) Two method for determining coupon package: off-line (next visit) and on-line (time of sale). Coupon dispense through clerk or direct mail. (Col. 67, line 32 – col. 70, line 46. Fig. 17)</p>

	<p>3) system decide whether to award a coupon and what type of coupon to award. e.g. If shopper consistently does not buy from deli, generate coupon to induce customer to shop at the deli. Stored data (shopping history can also be used to generate targeted coupon. (Col. 70, line 47 – col. 73, line 48. Fig. 18)</p>
--	---

U.S. 6,067,525	Deaton – U.S. 5,201,010
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, col. 6, lines 4-24. A method and system is disclosed for performing targeted marketing on infrequent shoppers. The system collects and accumulates selected additional transactional data, thus allowing the store to adopt risk management approach to check verification tailored to the store's particular customer and financial situation, and allowing the store to develop customer profiles and to target advertising, marketing and promotions, and otherwise improve customer relations.
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Check transaction Processing system (CTPS, col. 8, line 47- col. 20, line 31) Targeted Marketing col. (59 line 25 –col. 78, line 48)
an event manager, coupled to the subsystems and configured to	CTPS contains multiple subsystems including an Event Manger (col. 50, line 32 – col. 56, line 19)
detect one or more changes in state characteristic of an event occurring within the system,	Event driven activities (col. 31, line 51- col. 32, line 32).
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	1) Marketing based on range of last shopping dates (Col. 66, line 31 – col. 67, line 55. Fig. 16) Event – sale with a check / check processing Context – customer's last shopping date / customer's shopping history 2) Dissemination of Point of Sale coupons and direct mail coupons based upon shopping history (Col. 67, line 32 – col. 70, line 46. Fig. 17) Event – sale with a check / check processing Context – whether the utosome is a high volume shopper, primary shopper, or secondary (infrequent) shopper 3) Dissemination of point of sale coupons and direct mail coupons based upon scanned data (Col. 70, line 47 – col. 73, line 48. Fig. 18) Event – sale with a check / check processing Context – particular store departments in which the products being purchased
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	1) If this customer's last shopping date falls within the range of preselected shopping date range write this record to target file. Selection may also include a minimal dollar amount in a preselected time range. A promotion may then be selectively offered by the retail establishment to customers within the second database. (Col. 66, line 31 – col. 67, line 55. Fig. 16) 2) If secondary shopper dispense coupon value A (high incentive to make secondary shopper primary shopper) If primary shopper dispense coupon value B (lesser incentive package to maintain consistency) If high volume shopper dispense coupon value C (high value to hold on to especially good shopper) Two method for determining coupon package: off-line (next visit) and

	<p>on-line (time of sale). Coupon dispense through clerk or direct mail. (Col. 67, line 32 – col. 70, line 46. Fig. 17)</p> <p>3) system decide whether to award a coupon and what type of coupon to award. e.g. If shopper consistently does not buy from deli, generate coupon to induce customer to shop at the deli. Stored data (shopping history can also be used to generate targeted coupon. (Col. 70, line 47 – col. 73, line 48. Fig. 18)</p>
<p>automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.</p>	<p>1) If this customer's last shopping date falls within the range of preselected shopping date range write this record to target file. Selection may also include a minimal dollar amount in a preselected time range. A promotion may then be selectively offered by the retail establishment to customers within the second database. (Col. 66, line 31 – col. 67, line 55. Fig. 16)</p> <p>2) If secondary shopper dispense coupon value A (high incentive to make secondary shopper primary shopper) If primary shopper dispense coupon value B (lesser incentive package to maintain consistency) If high volume shopper dispense coupon value C (high value to hold on to especially good shopper) Two method for determining coupon package: off-line (next visit) and on-line (time of sale). Coupon dispense through clerk or direct mail. (Col. 67, line 32 – col. 70, line 46. Fig. 17)</p> <p>3) system decide whether to award a coupon and what type of coupon to award. e.g. If shopper consistently does not buy from deli, generate coupon to induce customer to shop at the deli. Stored data (shopping history can also be used to generate targeted coupon. (Col. 70, line 47 – col. 73, line 48. Fig. 18)</p>

U.S. 6,067,525	Johnson – U.S. 5,283,865
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, . A computerized system provides a salesperson with assistance related to training and sales of parts corresponding to particular products.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 6, ln 15-25. The computer system 100 incorporates a display apparatus 102, a data storage device 104, a part selection device 106 and a user interface mechanism 108. The display apparatus 102 preferably is operatively interconnected to the data storage device 104 and the user interface mechanism 108. See Col. 7, ln 8-30. The computer system 100 may further be enhanced by including a report generation apparatus 100, shown in FIG. 1, and by storing part identity information 136 and part prices 138 in data storage device 104. The computer system 100 may further include a report generation apparatus 110 which preferably is operatively interconnected to data storage device 104 and the user interface mechanism 108.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 6, ln 15-25. The data storage device 104 further preferably is operatively interconnected to the part selection device 106. In addition, part selection device 106 preferably is operatively interconnected to user interface mechanism 108. See Col. 6, ln 37-54. part selection device 106 provides for electronically selecting a particular part by navigating through a plurality of part choices menus.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See Col. 9, ln 6-28. Upon selecting the choice of specifying a battery by part number 200, a user is prompted to enter a part number 202. ... To select that particular part 208, part selection device 106 accesses the battery information 300 stored in data storage device 104 and shown in FIG. 6 to choose a particular part 250. By using the cross-reference file 294 in conjunction with the part number, a particular set of information is retrieved from battery information 300. Subsequently, the part selection device 106 ends the procedure get part by part number for battery 210. See Col. 10, ln 4-19. FIGS. 3A and 3B, after a user chooses to select a particular battery by equipment application 212, preferred part selection device 106 gets and displays section 1 information at 214. This section 1 information may be derived from all records in file format for battery cross-reference 270 which have a pointer 272 to info 1 data. The information may be derived by accessing file 270 and the first record containing a pointer 272 to info 1 data and subsequently accessing each record pointed to by a pointer 274 which points to the next record having a pointer to info 1 data.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the	See Col. 10, ln 37-col. 11, ln 49. . Subsequently, a user typically selects one of the elements from the section 2 information 220 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 224. If another

inferred context.	<p>section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 3 information 226 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user may select one of the elements from the section 3 information 228 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 230. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 4 information 232 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user typically selects one of the elements from the section 4 information 234 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 236. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 5 information 238 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user may select one of the elements from the section 5 information 240 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 242. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 6 information 244 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user typically selects one of the elements from the section 6 information 246 and part selection device 106 displays part information 248.</p> <p>After displaying the part information 248, part selection device 106 accesses the battery information 300 stored in data storage device 104 and shown in FIG. 6 to choose a particular part 250. By using the cross-reference file 294 in conjunction with the particular BCI group and designated cold cranking amps for the particular part, a particular set of information is retrieved from battery information 300. Subsequently, the part selection device 106 ends the get part by applications for battery procedure 252.</p>
-------------------	--

U.S. 6,067,525	Johnson – U.S. 5,367,627
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, . A computerized system provides a salesperson with assistance related to training and sales of parts corresponding to particular products.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 6, ln 18-28. The computer system 100 incorporates a display apparatus 102, a data storage device 104, a part selection device 106 and a user interface mechanism 108. The display apparatus 102 preferably is operatively interconnected to the data storage device 104 and the user interface mechanism 108. See Col. 7, ln 8-30. The computer system 100 may further be enhanced by including a report generation apparatus 100, shown in FIG. 1, and by storing part identity information 136 and part prices 138 in data storage device 104. The computer system 100 may further include a report generation apparatus 110 which preferably is operatively interconnected to data storage device 104 and the user interface mechanism 108.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 6, ln 18-28. The data storage device 104 further preferably is operatively interconnected to the part selection device 106. In addition, part selection device 106 preferably is operatively interconnected to user interface mechanism 108. See Col. 6, ln 40-57. part selection device 106 provides for electronically selecting a particular part by navigating through a plurality of part choices menus.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See Col. 9, ln 9-31. Upon selecting the choice of specifying a battery by part number 200, a user is prompted to enter a part number 202. ... To select that particular part 208, part selection device 106 accesses the battery information 300 stored in data storage device 104 and shown in FIG. 6 to choose a particular part 250. By using the cross-reference file 294 in conjunction with the part number, a particular set of information is retrieved from battery information 300. Subsequently, the part selection device 106 ends the procedure get part by part number for battery 210. See Col. 10, ln 7-22. FIGS. 3A and 3B, after a user chooses to select a particular battery by equipment application 212, preferred part selection device 106 gets and displays section 1 information at 214. This section 1 information may be derived from all records in file format for battery cross-reference 270 which have a pointer 272 to info 1 data. The information may be derived by accessing file 270 and the first record containing a pointer 272 to info 1 data and subsequently accessing each record pointed to by a pointer 274 which points to the next record having a pointer to info 1 data.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the	See Col. 10, ln 40-col. 11, ln 52. . Subsequently, a user typically selects one of the elements from the section 2 information 220 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 224. If another

inferred context.	<p>section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 3 information 226 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user may select one of the elements from the section 3 information 228 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 230. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 4 information 232 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user typically selects one of the elements from the section 4 information 234 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 236. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 5 information 238 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user may select one of the elements from the section 5 information 240 and the part selection device 106 checks if other sections are needed to particularly define the equipment application 242. If another section level is not needed to completely define the battery for a particular equipment application, part selection device 106 displays part information 248. Otherwise, part selection device 106 will get and display section 6 information 244 from file 270 records in a similar manner as was used to get and display section 1 information.</p> <p>Subsequently, a user typically selects one of the elements from the section 6 information 246 and part selection device 106 displays part information 248.</p> <p>After displaying the part information 248, part selection device 106 accesses the battery information 300 stored in data storage device 104 and shown in FIG. 6 to choose a particular part 250. By using the cross-reference file 294 in conjunction with the particular BCI group and designated cold cranking amps for the particular part, a particular set of information is retrieved from battery information 300. Subsequently, the part selection device 106 ends the get part by applications for battery procedure 252.</p>
-------------------	--

U.S. 6,067,525	Cherrington – U.S. 5,717,595
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, . See drawing figure 1 and column 4, lines 65-66. an integrated highly automated vehicle diagnosis, estimating, and invoicing system 10
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 4, ln 66-col. 5 ln 4. A technician terminal 12 is coupled to a measurement and instructions printer 14 and to a digital measuring instrument 16. A recommended/suggested services printer 18 is also coupled to the technician terminal 12, as is a point of sale terminal 20, and a point of sale printer 22. See Col. 5, ln 16-19. Also coupled to the technician terminal 12 are a customer/inspection database 24, a measurements/specifications database 26, a shop manual database 28, a parts catalog database 30, and an inspection guidelines database 32.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. Inspection program. This inspection is performed using the digital measuring instrument 16 The measurement is input into the "brake measurements" inspection screen generated by the inspection program. The technician inputs the make, model and year of the vehicle being inspected prior to conducting the inspection.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. The inspection program accesses the measurements/specifications database 26 in order to determine the configuration of braking equipment for the vehicle being inspected. As a result, the inspection program will only request measurements for rotors and/or brake drums as appropriate for the particular make, model and year of vehicle on which the inspection is being conducted. The inspection guidelines retrieved from the inspection guidelines database 32 are automatically selected based on the particular inspection screen at which "Uniform Inspection Guidelines" is selected. In this way, the technician is able to view context-sensitive inspection guidelines for the area of the vehicle being inspected without the need for the technician to leave the inspection area to consult printed manuals containing inspection guidelines.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. Determination as to whether a particular service is recommended or required is made automatically by the inspection program based on the inspection results, and the inspection guidelines retrieved from the inspection guidelines database 32, and measurements and specifications retrieved from the measurements/specifications database 26. For example, required services may include those

	<p>services or parts which relate to aspects of the inspection which were out of specification or tolerance. The suggested services, in contrast, may relate to those aspects of the inspection which indicate parts or services that are still within specification, but that are within a prescribed tolerance, i.e., a percentage, e.g., fifteen percent, or a prescribed amount of being out of specification. For example, when brake pad measurements are taken, if the brake pads have less than one or two thirty-seconds of an inch thickness, they are suggested for replacement. Each suggested or required service or part repair/replacement is automatically accompanied by a detailed standardized explanation of the "condition" selected by the technician during the inspection.</p> <p>Along with the printing of the recommended/suggested services report, the inspection report is communicated to the point of sale terminal 20, which is modified with a point of sale program.</p>
--	---

U.S. 6,067,525	Cherrington – U.S. 5,657,233
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, . See drawing figure 1 and column 4, lines 65-66. an integrated highly automated vehicle diagnosis, estimating, and invoicing system 10
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 4, ln 66-col. 5 ln 4. A technician terminal 12 is coupled to a measurement and instructions printer 14 and to a digital measuring instrument 16. A recommended/suggested services printer 18 is also coupled to the technician terminal 12, as is a point of sale terminal 20, and a point of sale printer 22. Col. 5, ln 16-19. Also coupled to the technician terminal 12 are a customer/inspection database 24, a measurements/specifications database 26, a shop manual database 28, a parts catalog database 30, and an inspection guidelines database 32.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. Inspection program. This inspection is performed using the digital measuring instrument 16 The measurement is input into the "brake measurements" inspection screen generated by the inspection program. The technician inputs the make, model and year of the vehicle being inspected prior to conducting the inspection.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. The inspection program accesses the measurements/specifications database 26 in order to determine the configuration of braking equipment for the vehicle being inspected. As a result, the inspection program will only request measurements for rotors and/or brake drums as appropriate for the particular make, model and year of vehicle on which the inspection is being conducted. The inspection guidelines retrieved from the inspection guidelines database 32 are automatically selected based on the particular inspection screen at which "Uniform Inspection Guidelines" is selected. In this way, the technician is able to view context-sensitive inspection guidelines for the area of the vehicle being inspected without the need for the technician to leave the inspection area to consult printed manuals containing inspection guidelines.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See column 7, lines 36-67, column 8, lines 1-67 and column 9, lines 1-16. Determination as to whether a particular service is recommended or required is made automatically by the inspection program based on the inspection results, and the inspection guidelines retrieved from the inspection guidelines database 32, and measurements and specifications retrieved from the measurements/specifications database 26. For example, required services may include those services or parts which relate to aspects of the inspection which

	<p>were out of specification or tolerance. The suggested services, in contrast, may relate to those aspects of the inspection which indicate parts or services that are still within specification, but that are within a prescribed tolerance, i.e., a percentage, e.g., fifteen percent, or a prescribed amount of being out of specification. For example, when brake pad measurements are taken, if the brake pads have less than one or two thirty-seconds of an inch thickness, they are suggested for replacement. Each suggested or required service or part repair/replacement is automatically accompanied by a detailed standardized explanation of the "condition" selected by the technician during the inspection.</p> <p>Along with the printing of the recommended/suggested services report, the inspection report is communicated to the point of sale terminal 20, which is modified with a point of sale program.</p>
--	---

U.S. 6,067,525	Johnson – U.S. 6,023,683
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, An electronic sourcing system. column 3, lines 49-56,
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See column 4, lines 1-5; 13-20. a local computer 20, Electronic sourcing system 5 also includes a requisition/purchasing system 40, preferably but not necessarily the Fisher RIMS system, and a search program 50 that is capable of searching through large volumes of information quickly and accurately. Fisher RIMS system 40 is comprised of numerous program modules, Fisher RIMS system 40 also includes several Fisher RIMS databases 42.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See column 4, lines 61-66. Where the Fisher RIMS system is in use with electronic sourcing system 5, a host computer 10 located at a Distributor site is also provided, as shown in FIG. 1A. Host computer 10 controls all inventory, pricing and requisitioning operations of the Distributor's regularly stocked items using host pricing and inventory databases 11. See column 5, lines 18-38. Interface 60 is also a part of electronic sourcing interface system 5. Interface 60 communicates shared data between requisition/purchasing system 40 and search program 50. As shown in FIG. 2, interface 60 preferably includes three linking programs to interface requisition/purchasing system 40 and search program 50. A typical data exchange may begin with requisition/purchasing system 40 (which, in the illustrated embodiment, is the Fisher RIMS system) requesting information from catalog database 36 via search program 50. Once a search by search program 50 has been completed, the selected information will be communicated to requisition/purchasing system 40 via interface 60. Alternatively, if the search of catalog database 36 is initiated from search program 50, the information selected from the search is returned to requisition/procurement system 40 via interface 60.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See column 17, lines 23-67 and column 18, lines 1-29. When a customer asks for products by manufacturer part number or a competitor's catalog number, the CSR has access to cross-reference files, as earlier described, either maintained on the local host or maintained on the Distributor host computer 210. Appropriate Distributor catalogs and manufacturer catalogs then are consulted, using TV-2 search program 250 and proper selection of Distributor catalogs and of catalogs and bulletins from manufacturers whose products Distributor regularly sells. Catalogs and bulletins are contained in catalog database 236.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See column 17, lines 23-67 and column 18, lines 1-29. The resultant lists of products are then transferred by Shell program 252 to a work-in-progress requisition 260, and then entered from graphical user interface 254 directly onto Distributor's mainframe computer 210 as orders from the applicable customer to Distributor. The CSR, knowing which items are available from which Distributor warehouse and direct-shipping supplier, then may divide the customer's requested items into multiple orders, so as to assure that each order is completely filled by a single shipment.

U.S. 6,067,525	Wollaston – 6,061,506
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, See drawing figure 1 and column 10, lines 30-51, Computerized Smart System (2). An adaptable user-defined strategy-based computerized system for emulating the continuous operation of a business scenario. Car dealership example.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See column 11, lines 1-11, plurality of interrelated activities and associated events.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 4, lines 23-65. The present invention provides an automated computer system which continuously and quiescently operates user-defined business procedures and activities with minimal human intervention. The present invention recognizes that all businesses and organizations are operated by the performance of specific cyclical activities. A smart system contemplated under the present invention tracks all predefined pending activities and events, storing and retrieving appropriate information in a comprehensive set of related databases. Also provided is a unique Update Fields mechanism for maintaining and changing data in individual knowledge base records on the basis of dynamically changing circumstances, with such circumstances being anticipated and predefined within the knowledge base. See column 12, lines 56-67, column 13, lines 1-67, column 14, lines 1-67 and column 15, lines 1-24.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See column 10, lines 30-65, Knowledge Base (5). The diversity of information stored in knowledge base 5 enables the structure and day-to-day operations of a business organization to be emulated and enables such comprehensive and timely business-related information to be instantaneously accessed by authorized personnel by computer system 40 comprising processing module 8 which may be invoked in either an automatic or a manual mode.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See Col. 5, lines 5-26. The present invention, either through automatic operation or responding to user requests, performs output functions such as generating letters, sending faxes, issuing instructions, etc., which are required to continuously operate the business and the like. See column 12, lines 56-67, column 13, lines 1-67, column 14, lines 1-67 and column 15, lines 1-24.

U.S. 6,067,525	Moore – U.S. 5,630,127
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, Col. 2, ln 15-25. The invention can be applied to many types of rule-based (or even expert) systems for any type of industry or vertical market.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 3, ln 40-col. 4, ln 6. The present invention was developed for use in a Global Risk Management System (GRMS). GRMS interfaces with operational systems and management information systems. Operational systems provide GRMS with business event data in the form of transactions or balance updates. Management information systems provide GRMS with reference data such as: product, organization, and economic data. GRMS also interfaces with market data providers to obtain information such as, for example: foreign exchange rates, market prices and counter party ratings.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 4, ln 28-44. GRMS 108 is an event driven system. All significant inputs are treated as events that GRMS must process and analyze to produce information for exposure and risk management. Files sent to GRMS from external sources will also be processed into discrete events.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See column 1, lines 38-65, column 2, lines 26-55, column 4, lines 36-44 & 52-60 and column 5, starting at line 31. Data and rules will be stored in the GRMS database 102. The data will be stored as a relational database, preferably utilizing the IBM database DB2 or other comparable relational database, and are part of the CMIS database 102. Rules for GRMS will be stored as objects in the database. The term "object" is an abstraction (like a variable in mathematics) that represents a value that is returned by an associated retrieval program.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See column 4, lines 62-67. Once the calculation is performed in GRMS, the result of the calculation is passed to a result handler. The result handler will also have rules to follow for the disposition of the result. Results can cause an update to the CMIS database 102, a long report to be created and routed to one or many people of the organization, or no further action.

U.S. 6,067,525	Dalnekoff – U.S. 4,931,932
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, col. 3, ln 61-col. Ln 7. While the present invention will be described in the context of an airline reservation system, it should be recognized that its potential uses extend to all applications where reservations are made, items are bought and sold via computer interaction, or transactions are presented to a computer system.</p>
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	<p>See Figure. Col. 4, ln 8-59.</p> <p>one or more transaction forwarding and storage stations 12, which function as input/output stations, inventory storage device(s) 14, 14' for storing an inventory of items to be sold</p>
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	<p>See Figure. Col. 60, ln 8-col. 5, ln 34.</p> <p>central processing unit 16 and the information processing unit 20 each may comprise any special or general purpose computer known in the art including micro-computers. Each one may be programmed to carry out the aims of the present invention or may contain suitable circuitry to do the same.</p> <p>In operation, the information processing unit 20 is the heart of the system. It receives requests from a user and through suitable programming and/or circuitry evaluates the requested transaction to determine if it can be successfully or unsuccessfully completed.</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>See col. 5, ln 11-46. If a requested inventory item is available, the unit 20 may effect a sale of the item or make a reservation thereof. At the same time, the unit 20 may instruct the unit 16 to change the status of the item within the storage device(s) 14, 14' from available to unavailable.</p> <p>The unit 20 may also be used to generate an inventory of computer transactions and to either immediately process the transactions or store the inventory for later processing.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>See col. 5, ln 47-col. 6, 48.</p> <p>After a list or inventory of requests or transactions has been generated or compiled, the unit 20 through its programming and/or circuitry may be used to continuously and automatically query an inventory storage system about the availability of requested items or the transactions on the list. This search or query function may continue until one or more items in the inventory have been sold or an operator intervenes and stops the search function. When a previously uncompleted transaction is completed such as the sale of a wait-listed seat, the processing unit 20 automatically clears it from the list of requests.</p> <p>The system 10 is further desirable in that it may be used for non-conventional reservation booking techniques. For example, the</p>

	<p>system may be used to define a set of standard itineraries which are then stored in either the memory 26 of the central processing unit or the memory 21 of the unit 20. To make reservations and/or sell seats in this mode, the agent inputs the name of the person taking the trip and the appropriate dates of travel and the unit 16 or 20 conforms the request to the standard itineraries. Alternatively, the unit 20 could be used to formulate a request in accordance with predefined criteria and then query the inventory storage system about the availability of the requested item.</p>
--	---

	<p>The system may also be used for non-traditional booking methods such as: (1) off-line interfaces for high volume transactions; (2) off-line interfaces for non-travel professionals; (3) off-line interfaces which are part of other office automation systems; or (4) expert systems and other heuristic or non-heuristic methods for determining optimal routing.</p>
--	--

U.S. 6,067,525	Lockwood – U.S. 5,309,355
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, col. 4, ln 4-50. It will be understood that such a system may be used in a variety of other service-oriented industries, such as the retail sales and real estate industry, various financial services and the like.</p>
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	<p>See Col. 4, ln 51 – col. 6, ln 12.</p> <p>The system basically comprises one or more special information and sales terminals 2 linked to an airline computerized reservation system 1 which gives access to the data processing installations of various travel suppliers 4.</p> <p>The information and sales terminals 2 include one or more audio-visual data sources 9.</p>
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	<p>See Col. 5, ln 13 – col. 6, ln 19.</p> <p>The basic selection data which is either entered on the keyboard 13 by the operator or read by the memory card reader 12 are loaded into the input registers 15 of the microprocessor 14. This basic selection data includes the type of service requested (such as ski weekend, cruise, or camping trip), the approximate date of departure and return, the destination and customer characteristics such as age, gender and preferences.</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>See Col. 5, ln 13 – col. 6, ln 19.</p> <p>REMOTE 30 represents information which is accessed from the computerized reservation system 1. This information; airline flight times 32 and hotel availability 33 is critical to the tour sales presentation if transportation or lodging is required. Therefore, LOCAL 20 and REMOTE 30 act in conjunction to present an integrated and individualized travel and tour sales travelogue. REMOTE 30 creates dynamic presentations of transitory information such as weather conditions 35 or currency exchange rates 37.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>See Col. 5, ln 13 – col. 6, ln 19.</p> <p>This information serves as addresses for a programmable read only memory (PROM) 16 or other similar device which is driven by a sequencer 17 to deliver a series of specific disc segment addresses 18 for the videodisc memory data source 9. Some of the addressed segments on the videodisc correspond to inquiries 19 which are sent via a modem 11 to the airline reservation system 1. The answers, mostly reservation information 30, when received, are presented on a split-screen or recorded on the erasable optical disc or other suitable medium for display as part of the sales presentation on the CRT 8.</p> <p>the PROM performs automatically and almost instantly, the sequential and time-consuming information gathering and organization that a travel agent must process using conventional reservation and information sources.</p>

	<p>See Col. 7, ln 63-col. 8, ln 30.</p> <p>This tour creating flexibility allows the computerized reservation system 1 to market `special` or discounted tours with short-term availability.</p>
--	--

U.S. 6,067,525	Bosco – U.S. 5,191,522
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, The invention also provides, in a specific embodiment an information storage, processing and reporting system which integrates the sales, underwriting, administration, claims and actuarial functions that support and service the sale and administration of group insurance products.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See figure 11, 12. Col. 22, ln 8-31. The access architecture of the integrated group insurance system according to the invention, the integrated system is composed of the sales 111, underwriting 112, administration 113, and actuarial 114 Workstation sub-systems. Each of the functional workstation sub-systems 111-114 is comprised of a plurality of the functional program modules depicted in FIG. 11. Certain of the functional program modules may be accessible by a plurality of the functional workstations. Each of these functional program modules is connected with the single Group Insurance Relational Business Data Base 115, which contains all of the data concerning each account or case and is structured in accordance with the data model described above. The Group Business Relational Data Base 115, may be accessed from any of the integrated workstations FIG. 13, which are part of the overall system and may be used with any of the sub-system program modules 70-109.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 23, ln 31-44. The Sales Tracking program module 70 compiles initial pre-sale data about an insurance account or case, its producer, coverage and competitors and allows multiple users on-line access to the data. Data on Renewals, cancellations and additions to existing policies are also tracked with this module. This module 70 tracks the case history with the Insurer from initial contact until the case is sold. Module 70 also generates sales management reports. Some of the initial data compiled and processed on the System on module 70, is electronically transferred to module 71 once the case is sold. The arrows in FIG. 11 show the electronic transfer of compiled and processed data from one program module to another.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	See col. 23, ln 55 –col. 24, ln 20. Using case and census information, the Rating program module 73 will generate manual rates. This function is currently most often contained in stand-alone systems in the insurance underwriting departments. The Module 73 calculates rates for both new and renewal business.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	See col. 23, ln 55 –col. 24, ln 20. Sales proposals are generated by the Proposals module 74 following a request to quote from a producer or client. The sales proposal outlines the coverages, rates, and restrictions of the Insurer's quote. Custom proposals may be generated by the module

	<p>74 to suit individual needs.</p> <p>After a sale and an account is issued, the Case Management program module 75 performs the day-to-day maintenance of the case information. Changes can be made to the Case Management 75 central data source and all departments may inquire instantaneously and obtain current information reflecting the changes as they are made. Automation of this function reduces field telephone inquiries by offering on-line access to case data and status. The Case rates are maintained and updated as needed and provide a direct feed of such data to the billing modules 78, 79, 80, 81 and 82.</p>
--	---

U.S. 6,067,525	Miller – U.S. 5,446,653
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, col. 5, lines 30-49. In the illustrated embodiment, insurance policies are built from a software library of coverage provisions that can be rearranged and used in any number of ways according to the needs and coverage preferences of a proposed insured.</p>
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	<p>See col. 5, line 50 – col. 6, line 21.</p> <p>A main processor 13, such as a mainframe computer, interfaces with a user through a terminal 11. Processor 13 is coupled to conventional memory (e.g., magnetic disk storage) that stores a library of standard insurance policy clauses 17 together with insurance policy rule sets 19.</p> <p>Printing of the final policy is provided by a print system 15 coupled to main processor 13.</p>
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	<p>See col. 6, line 22 – 51.</p> <p>In order to create an insurance policy, a user selects one or more desired coverages from a list of coverages displayed at terminal 11. For example, in building a specialty marine insurance policy, the user may be offered, via a menu provided on terminal 11, a choice of property coverages such as Builders' Risk, Computerized Business Equipment, Contractor's Equipment, Installation, and Scheduled Property. The user will also input the name of the proposed policy holder, the policy holder's state of domicile, the effective dates of the policy, and other relevant coverage information.</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>See col. 6, line 22 – 51.</p> <p>In response, main processor 13 will test each insurance policy clause rule set provided in library 19 to determine which of the rule sets are satisfied by the entered coverage information. Each of the rule sets contains information identifying which insurance policy clause it corresponds to. The insurance policy clauses that correspond to the rule sets satisfied by the entered coverage information are listed by clause numbers. The endorsement rule sets are then tested to determine which are satisfied by the listed insurance policy clauses. A list of the applicable endorsements is then presented to the user, who selects the endorsements which are desired. The desired endorsements are used to replace insurance policy clauses, and a policy record containing a list of the remaining insurance policy clauses and the selected endorsements is generated for use by the print system in printing the final insurance policy.</p> <p>See col. 8, line 58 – col. 9, line 64.</p> <p>After the endorsements have been selected, the insurance policy rule sets are formally processed at box 20 to tentatively select the insurance policy clauses that are required for the coverages selected by the user.</p>

automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.

See col. 13, line 53 – col. 14, line 32.

The document creation routine of FIG. 5 commences at box 170, and at box 172 the policy record file 70 is retrieved. Each clause number contained in the policy record file is processed one at a time, as indicated at box 174. An end-of-file determination is made at box 176 to determine if all of the clauses in the policy record have been processed.

U.S. 6,067,525	Kawashima – U.S 5,168,445
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation. Nonetheless, see Abstract, An automated ordering system in a retail shop adapted to automatically order goods.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	See Col. 3, lines 7-26. The automatic ordering system is constructed as shown, in block form, in FIG. 1. Thus, this system comprises a terminal 1 through which input/output of information is effected by the worker; a variable condition setter 2 for setting variable condition data 10 indicative of factors of change; a sales volume predictor 3, responsive to variable condition data 10, POS data 11 and a correction rule table 15 for storing correction rules used to correct the sales volume, for predicting the volume of sales; an ordering amount calculator 4, responsive to predictive data 12 produced from the predictor 3, for calculating the amount of orders; a rule modifier 6 for modifying the correction rule table 15 and a diagnostic rule table 16 used to diagnose the condition of calculated ordering data 14 and stock data 13; a condition diagnostic unit 7 for diagnosing the condition in accordance with the diagnostic rule table 16; and an ordering processor 5, responsive to a diagnostic message 17 produced from the diagnostic unit 7 and ordering data 14, for delivering an order slip 18.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	See Col. 3, lines 29-col.4, line 2. The worker inputs information indicative of events in the market area and goods on sale at his own shop. The information is applied via a signal line 104 to the variable condition data 10, to either input new data representative of the variable condition valid for a few days which is necessary for predicting the volume of sales, or change the contents of the variable condition data (step 201). the variable condition setter 2 fetches the data and variable condition corresponding to the inputted setting date 401 from the variable condition data 10 via a signal line 103 so as to display it on the variable condition setting screen. Items of information about weather 306, place 307 and name 308 of an event, name 309 and selling status (for example, shop closing and bargain sale) 310 of other shops, goods 311 on sale at the shop manager's own (for example, chocolate, cookie and curry) and trade name 312 are displayed at a selective item table 313 on the variable condition setting screen. The worker selects information items from the selective item table 313 and inputs them. The inputted information is then supplied via a signal line 102 to the variable condition setter 2 which in turn updates the contents of the variable condition data 10 or inputs new data thereto via the signal line 104. See Col. 4, line 3-col.5, line 17. The sales volume predictor 3 is then supplied with the updated or newly inputted contents of variable condition data 10 via a signal

	line 105, POS data 11 collected at a POS terminal via a signal line 106 and correction rule table 15 via a signal line 107 and it performs calculation to predict the sales volume.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>See Col. 5, line 18 – col. 7, line 43.</p> <p>The condition diagnostic unit 7 is operable to perform the following procedure. Firstly, the condition diagnostic unit 7 is supplied with POS data 11, stock data 13 and diagnostic rule table 16 via signal lines 115, 117 and 126, respectively. Next, it analyzes the goods "tendency to sell" in order to know whether the volume of sales of individual goods groups and individual goods increases or decreases, by applying the mathematical method of "least squares" to the sales volume in POS data, with respect to individual goods groups and individual goods.</p> <p>The condition diagnostic unit diagnoses to provide various diagnostic messages which are stored in the diagnostic message 17.</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>See Col. 5, line 18 – col. 7, line 43.</p> <p>Subsequently, the ordering processor 5 fetches ordering data 14 and diagnostic message 17 via signal lines 112 and 132, respectively, and delivers an order amount and the diagnostic message to the terminal 1. The worker corrects the displayed order amount by locking up the diagnostic message. The ordering processor 5 fetches the corrected order amount via a signal line 130 to update the ordering data 14 and delivers an order slip 18.</p>

U.S. 6,067,525	Johnson – U.S. 5,349,662
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation.</p> <p>Nonetheless, see Abstract, Automatic detection of the activities of a user of a data processing system is provided by the use of an Activity Event Detection Process, an Activity Detection Process, and an Interrogator Process.</p>
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	<p>See FIG. 1; col. 5, lines 28-44.</p> <p>A data processing system 100 according to the present invention. The data processing system 100 includes a processor 02, which includes a central processing unit (CPU) 04, and a memory 06. Additional memory, in the form of a hard disk file storage 08 and a floppy disk device 10, is connected to the processor 02. Floppy disk device 10 receives a diskette 12 which has computer program code recorded thereon that implements the present invention in the data processing system 100. The data processing system 100 may include user interface hardware, including a mouse 14 and a keyboard 16 for allowing user input to the processor 02 and a display 18 for presenting visual data to the user. The data processing system 100 may also include a communications port 20 for communicating with a network or other data processing systems.</p>
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,	<p>See col. 5, lines 45-53.</p> <p>The User Activity Event Detection Process, illustrated in FIG. 2, detects user activity indicating events. The Activity Detection Process, illustrated in FIG. 3 and FIG. 4, evaluates the user activity indicating events to determine the activity of a user. See col. 6, 19-38.</p> <p>The events may be described or represented by parameters, including, but not limited to, interrupt levels, interrupt priorities, semaphore content, or interprocess communication content. The parameters are sufficient to discriminate a user activity indicating event from events which do not indicate the activity of a user. For example, the User Activity Event Table may include a description of an event such as a telephone line off-hook signal, indicating that the user has answered his phone. The User Activity Event Table may also include a description of an event such as the opening of an electronic mail document, indicating that the user is reading the document. The User Activity Event Table may also include a description of an event such as the invocation of a spreadsheet program, indicating that the user is working on a spreadsheet. However, the User Activity Event Table does not include any events which are not initiated by a user activity, such as the automatic start or termination of programs by the data processing system.</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on	<p>See col. 9, line 36 –col. 10, line 54.</p> <p>Interrogator Process portion of the present invention</p> <p>Processing block 340 provides to the requesting user the return</p>

the detected changes in state, and	status and other results of the query as returned from the Activity Detection Process. Embodiments of the return status and other results include status strings or status codes which may be mapped to local user friendly interfaces.
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>See col. 11, line 7 – 59.</p> <p>Thereafter, process block 445 compares the sequence of user activity indicating events in the extracted entry to the user activity indicating events detected in the specified time interval.</p> <p>Thereafter, decision block 450 determines if the comparison of process block 445 yields a sequence match in which all user activity indicating events in the extracted entry are contained in the detected user activity indicating events.</p> <p>If the comparison of process block 445 yields a sequence match, then a user activity in the entry corresponding to the sequence is recognized. Thereafter, process block 465 calculates a time interval of the recognized user activity by computing a time interval between time stamps of events designated in the sequence. For a recognized user activity still in progress, process block 465 calculates the time interval of the recognized user activity by computing the time interval between the time of inquiry and a time stamp of an event designated in the sequence. Thereafter, process block 470 returns a message describing the recognized user activity, the start time of the recognized user activity based on a time stamp of a first designated event, and the time interval of the recognized user activity calculated by process block 465.</p>

U.S. Pat. No. 6,067,525 Claim 20	Harrison, H.C. & Qizhong, Gong: <i>An Intelligent Business Forecasting System</i>, Association of Computer Machinery 089791-558-5, 1993
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>“Integration of expert systems technology with a forecasting decision support system;” - “A rule-based expert system is integrated to assist the user in selecting appropriate models based on the user’s requirements and data patterns.” (Abstract)</p> <p>Development of a intelligent business forecasting system assists in selecting appropriate models based on users’ requirements an data patterns. (§1.1)</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process,	<p>“Monitor manages the system resources and acts as a scheduler of the system process.” (§3.4)</p> <p>“For example, if the user selects ‘expert advice’ from the main menu, the system asks: Do you want the system to check the pattern of your data?” (§3.4)</p>
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	<p>In response to the system question to user: “Do you want the system to check the pattern of your data?, the user can either let the expert system do the data analysis or enter information about the data pattern. After the expert system suggests the appropriate models, the user has three options:...” (§3.4)</p>
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	<p>“The monitor organizes the forecasting functions in a systematic manner. It supports the comparing and combining of up to 5 different models, incorporating judgment forecasting, residual analysis, and monitoring forecasting procedures.” (§3.4)</p>

U.S. Pat. No. 6,067,525 Claim 1	Stone, Robert W. & Good, David J.: <i>Expert Systems and Sales Strategies</i>, Association of Computer Machinery 089791-416-3, 1990
<p>1. A computer implemented sales system used to facilitate a sales process, the system comprising:</p>	<p>The preamble is not a limitation.</p> <p>The rapid growth of artificial intelligence, and more specifically expert systems, in business has generated a need for investigations concerning computer based intelligence systems within all facets of a business organization.” (Introduction, p.52)</p> <p>The focus of this paper is the application of one type of information technology (expert systems) to one point along the business value chain (marketing and sales). (The Theoretical Framework, p. 54)</p> <p>“The expert system provides a comprehensive method to alter marketing and sales strategies (e.g., adjust room rates or promote special packages). In the past, manual systems had no such comprehensive method. Using the expert system, any adjustments in strategy can quickly and comprehensively be made by altering the appropriate parameters in the expert system.” (Expert Systems and Sales Strategies, p. 55)</p>
<p>a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and</p>	<p>The focus of this paper is the application of one type of information technology (expert systems) to one point along the business value chain (marketing and sales). (The Theoretical Framework, p. 54)</p> <p>“The expert system provides a comprehensive method to alter marketing and sales strategies (e.g., adjust room rates or promote special packages). In the past, manual systems had no such comprehensive method. Using the expert system, any adjustments in strategy can quickly and comprehensively be made by altering the appropriate parameters in the expert system.” (Expert Systems and Sales Strategies, p. 55)</p>
<p>an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system,</p>	<p>Pages 54-55. When a returning guest calls the hotel to make a reservation, the reservationist enters the individual's name into the reservation system. Using the guest's name, the system produces a display containing the guest's previous stays and their preferences with respect to room characteristics. The reservationist then switches to a second 54 display which is a form to enter the guest's needs. Given the information regarding previous stays and current room preferences and each room's characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest's</p>

	preferences and needs while satisfying the hotel's management and sales strategies. Primary among these sales strategies is to minimize short term (one night) room vacancies.
inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	“Using the guest’s name, the system produces a display containing the guest’s previous stays and their preferences with respect to room characteristics. The reservationist then switches to a second display which is a form to enter the guest’s needs. Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest.” (Expert Systems and Sales Strategies, p. 54-55)
automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.	“Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest’s preferences and needs while satisfying the hotel’s management and sales strategies.” (Expert Systems and Sales Strategies, p. 55)

U.S. Pat. No. 6,067,525 Claim 20	Stone, Robert W. & Good, David J.: <i>Expert Systems and Sales Strategies</i>, Association of Computer Machinery 089791-416-3, 1990
<p>20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:</p>	<p>The preamble is not a limitation.</p> <p>The rapid growth of artificial intelligence, and more specifically expert systems, in business has generated a need for investigations concerning computer based intelligence systems within all facets of a business organization.” (Introduction, p.52)</p> <p>The focus of this paper is the application of one type of information technology (expert systems) to one point along the business value chain (marketing and sales). (The Theoretical Framework, p. 54)</p> <p>“The expert system provides a comprehensive method to alter marketing and sales strategies (e.g., adjust room rates or promote special packages). In the past, manual systems had no such comprehensive method. Using the expert system, any adjustments in strategy can quickly and comprehensively be made by altering the appropriate parameters in the expert system.” (Expert Systems and Sales Strategies, p. 55)</p>
<p>automatically detecting one or more changes in state characteristic of an event occurring in the sales process,</p>	<p>“When a returning guest calls the hotel...the reservationist enters the individual’s name into the reservation system. Using the guest’s name, the system produces a display containing ...” (Expert Systems and Sales Strategies, p. 54)</p>
<p>inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and</p>	<p>“Using the guest’s name, the system produces a display containing the guest’s previous stays and their preferences with respect to room characteristics. The reservationist then switches to a second display which is a form to enter the guest’s needs. Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest.” (Expert Systems and Sales Strategies, p. 54-55)</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	<p>Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest’s preferences and needs while satisfying the hotel’s management and sales strategies.” (Expert Systems and Sales Strategies, p. 55)</p>

U.S. Pat. No. 6,067,525 Claim 40	Stone, Robert W. & Good, David J.: <i>Expert Systems and Sales Strategies</i>, Association of Computer Machinery 089791-416-3, 1990
<p>40. A computer implemented sales system used to facilitate a sales process, the system comprising:</p>	<p>The preamble is not a limitation.</p> <p>The rapid growth of artificial intelligence, and more specifically expert systems, in business has generated a need for investigations concerning computer based intelligence systems within all facets of a business organization.” (Introduction, p.52)</p> <p>The focus of this paper is the application of one type of information technology (expert systems) to one point along the business value chain (marketing and sales). (The Theoretical Framework, p. 54)</p> <p>“The expert system provides a comprehensive method to alter marketing and sales strategies (e.g., adjust room rates or promote special packages). In the past, manual systems had no such comprehensive method. Using the expert system, any adjustments in strategy can quickly and comprehensively be made by altering the appropriate parameters in the expert system.” (Expert Systems and Sales Strategies, p. 55)</p>
<p>a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and</p>	<p>The focus of this paper is the application of one type of information technology (expert systems) to one point along the business value chain (marketing and sales). (The Theoretical Framework, p. 54)</p> <p>“The expert system provides a comprehensive method to alter marketing and sales strategies (e.g., adjust room rates or promote special packages). In the past, manual systems had no such comprehensive method. Using the expert system, any adjustments in strategy can quickly and comprehensively be made by altering the appropriate parameters in the expert system.” (Expert Systems and Sales Strategies, p. 55)</p>
<p>an event manager, coupled to the subsystems and configured to</p>	<p>Pages 54-55. When a returning guest calls the hotel to make a reservation, the reservationist enters the individual's name into the reservation system. Using the guest's name, the system produces a display containing the guest's previous stays and their preferences with respect to room characteristics. The reservationist then switches to a second 54 display which is a form to enter the guest's needs. Given the information regarding previous stays and current room preferences and each room's characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest's preferences and needs while satisfying the hotel's management and</p>

	sales strategies. Primary among these sales strategies is to minimize short term (one night) room vacancies.
detect one or more changes in state characteristic of an event occurring within the system,	“When a returning guest calls the hotel...the reservationist enters the individual’s name into the reservation system. Using the guest’s name, the system produces a display containing ...” (Expert Systems and Sales Strategies, p. 54)
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and	“Using the guest’s name, the system produces a display containing the guest’s previous stays and their preferences with respect to room characteristics. The reservationist then switches to a second display which is a form to enter the guest’s needs. Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest.” (Expert Systems and Sales Strategies, p. 54-55)
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest’s preferences and needs while satisfying the hotel’s management and sales strategies.” (Expert Systems and Sales Strategies, p. 55)
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	Given the information regarding previous stays and current room preferences and each room’s characteristics, the expert system suggests available rooms during the proposed stay of the guest. These suggestions are rooms which best fit the guest’s preferences and needs while satisfying the hotel’s management and sales strategies.” (Expert Systems and Sales Strategies, p. 55)

U.S. Pat. No 6,067,525 claim 1	Spezialetti, Madalene: <i>An Approach to Reducing Delays in Recognizing Distributed Event Occurrences</i>, Association of Computer Machinery 0-89791-457-0/91/0011/0155
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation.
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	Pages 156. At each processor at which a component of a particular event is located is an <i>event monitor</i> . This monitor will maintain information pertaining to the states of each component of the event which is located at that processor.
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system	<p>Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an event definition is monitored, and data pertaining to their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 156. "an <i>event definition</i> is a description of the activity which is to be detected by a monitoring system."</p> <p>Page 156. "An event definition, is assumed to be a predicate whose operands, or <i>components</i>, are combined or tested using relational, logical or temporal operators."</p> <p>Page 156. "In order to recognize an event occurrence, data regarding the states of the components must be collected and evaluated."</p> <p>Page 157. "Associated with each process is a <i>component monitor</i>, which is responsible for detecting changes to the monitored components within that process and, when an alteration occurs to a component, transmitting the value of that alteration to the appropriate event monitor."</p> <p>Page 162. "When a change occurs to a monitored component in a region, <i>r</i>, which may result in an occurrence of an event, the component's event monitor is informed of the charge."</p>
inferring occurrence of the event and a context in which the event	Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the

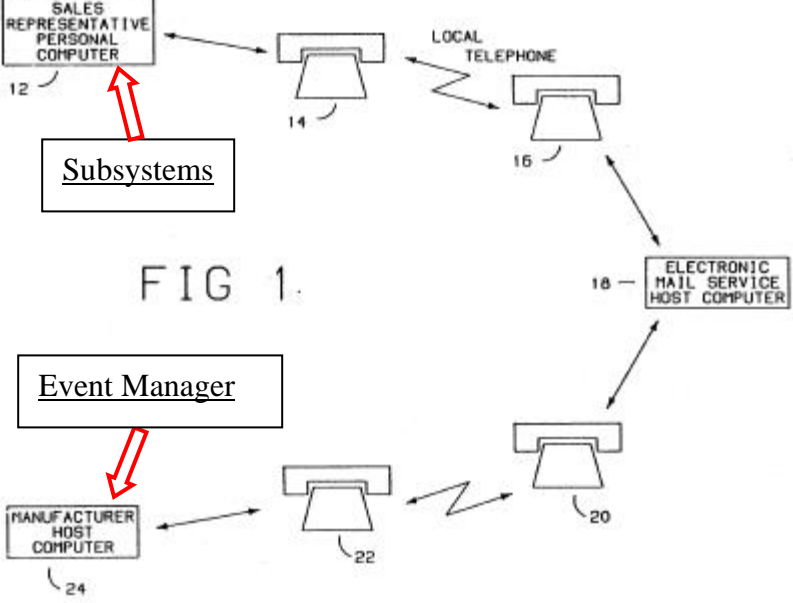
<p>occurred based at least in part on the detected changes in state, and</p>	<p>elements specified by an event definition is monitored, and data pertaining to their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 157. "An evaluation monitor may be assigned the task of evaluating an entire event or some portion of an event and is responsible for accumulating and organizing the data which is required to perform the evaluation and potentially make recognitions. ... [W]hen discussing the assignment of evaluation responsibilities, the module will be referred to as an evaluation monitor, although it may serve as an event monitor as well."</p> <p>Page 165. "[A] technique was presented to determine if an Immediate recognition of an event's occurrence could be guaranteed via the placement of evolution monitors based on the characteristics of an event's operators."</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	

U.S. Pat. No 6,067,525 claim 20	Spezialetti, Madalene: <i>An Approach to Reducing Delays in Recognizing Distributed Event Occurrences</i>, Association of Computer Machinery 0-89791-457-0/91/0011/0155
20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:	<p>The preamble is not a limitation.</p> <p>Nonetheless, Pages 156. At each processor at which a component of a particular event is located is an <i>event monitor</i>. This monitor will maintain information pertaining to the states of each component of the event which is located at that processor.</p>
automatically detecting one or more changes in state characteristic of an event occurring in the sales process;	<p>Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an event definition is monitored, and data pertaining to their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 156. "an <i>event definition</i> is a description of the activity which is to be detected by a monitoring system."</p> <p>Page 156. "An event definition, is assumed to be a predicate whose operands, or <i>components</i>, are combined or tested using relational, logical or temporal operators."</p> <p>Page 156. "In order to recognize an event occurrence, data regarding the states of the components must be collected and evaluated."</p> <p>Page 157. "Associated with each process is a <i>component monitor</i>, which is responsible for detecting changes to the monitored components within that process and, when an alteration occurs to a component, transmitting the value of that alteration to the appropriate event monitor."</p> <p>Page 162. "When a change occurs to a monitored component in a region, <i>r</i>, which may result in an occurrence of an event, the component's event monitor is informed of the charge."</p>
inferring occurrence of the event and a context in which the event occurred based at least	Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an event definition is monitored, and data pertaining to

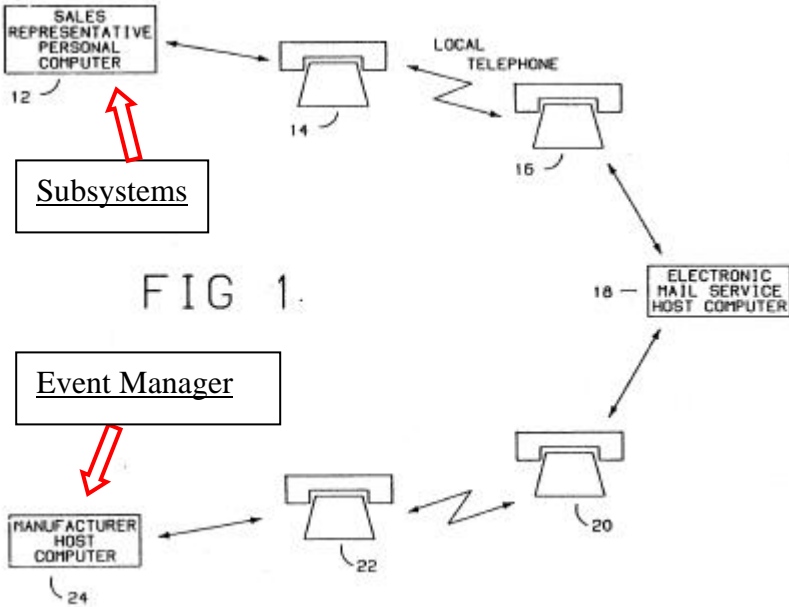
<p>in part on the detected changes in state; and</p>	<p>their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 157. "An evaluation monitor may be assigned the task of evaluating an entire event or some portion of an event and is responsible for accumulating and organizing the data which is required to perform the evaluation and potentially make recognitions. ... [W]henever discussing the assignment of evaluation responsibilities, the module will be referred to as an evaluation monitor, although it may serve as an event monitor as well."</p> <p>Page 165. "[A] technique was presented to determine if an Immediate recognition of an event's occurrence could be guaranteed via the placement of evolution monitors based on the characteristics of an event's operators."</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	

U.S. Pat. No 6,067, 525 claim 40	Spezialetti, Madalene: <i>An Approach to Reducing Delays in Recognizing Distributed Event Occurrences</i>, Association of Computer Machinery 0-89791-457-0/91/0011/0155
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation.
a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and	Pages 156. At each processor at which a component of a particular event is located is an <i>event monitor</i> . This monitor will maintain information pertaining to the states of each component of the event which is located at that processor.
an event manager coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring in the system,	<p>Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an event definition is monitored, and data pertaining to their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 156. "an <i>event definition</i> is a description of the activity which is to be detected by a monitoring system."</p> <p>Page 156. "An event definition, is assumed to be a predicate whose operands, or <i>components</i>, are combined or tested using relational, logical or temporal operators."</p> <p>Page 156. "In order to recognize an event occurrence, data regarding the states of the components must be collected and evaluated."</p> <p>Page 157. "Associated with each process is a <i>component monitor</i>, which is responsible for detecting changes to the monitored components within that process and, when an alteration occurs to a component, transmitting the value of that alteration to the appropriate event monitor."</p> <p>Page 162. "When a change occurs to a monitored component in a region, <i>r</i>, which may result in an occurrence of an event, the component's event monitor is informed of the charge."</p>
infer occurrence of the event and a context in which the event occurred based at least	Page 155. "One approach to aid in this analysis allows users to describe activity of interest in the form of event definitions. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an event definition is monitored, and data pertaining to

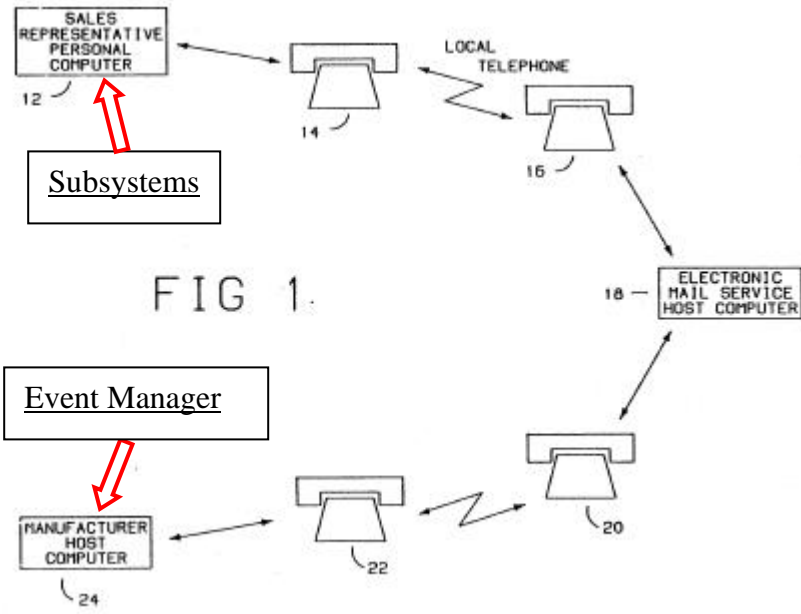
<p>in part on the detected changes in state,</p>	<p>their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event occurs at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is recognized at the point that the monitoring system detects its occurrence."</p> <p>Page 157. "An evaluation monitor may be assigned the task of evaluating an entire event or some portion of an event and is responsible for accumulating and organizing the data which is required to perform the evaluation and potentially make recognitions. ... [W]hen discussing the assignment of evaluation responsibilities, the module will be referred to as an evaluation monitor, although it may serve as an event monitor as well."</p> <p>Page 165. "[A] technique was presented to determine if an Immediate recognition of an event's occurrence could be guaranteed via the placement of evolution monitors based on the characteristics of an event's operators."</p>
<p>link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and</p>	
<p>automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.</p>	

U.S. Pat. No 6,067,525 claim 1	Long et al U.S. Pat. No. 5,117,354
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	The preamble is not a limitation, nonetheless col. 1 lines 7-13: “... present invention relates to systems for pricing and ordering goods... so that sales representatives can obtain pricing information, and place orders for the goods to be manufactured...”
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	 <p>Col. 3 lines 27-34:</p> <p>A variety of sales representatives in the field are each equipped with a personal computer 12 having processing capabilities, local memory and long term storage such as disk drives. Those personal computers 12 may be installed at the office locations of the sales representatives or may be portable units which they may carry with them to their home or other remote locations.</p> <p>Col. 2 lines 38-45:</p> <p>“...a central data processing facility connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative... an electronic mail serving host,... the manufacturing host...”</p>
an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system	<p>Col. 2 lines 38-45:</p> <p>“...a central data processing facility connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative... an electronic mail serving host,... the manufacturing host...”</p> <p>Co. 9 Lines 5-10:</p> <p>“...software embedded in step 80 [host] ...looks for mail placed in its</p>

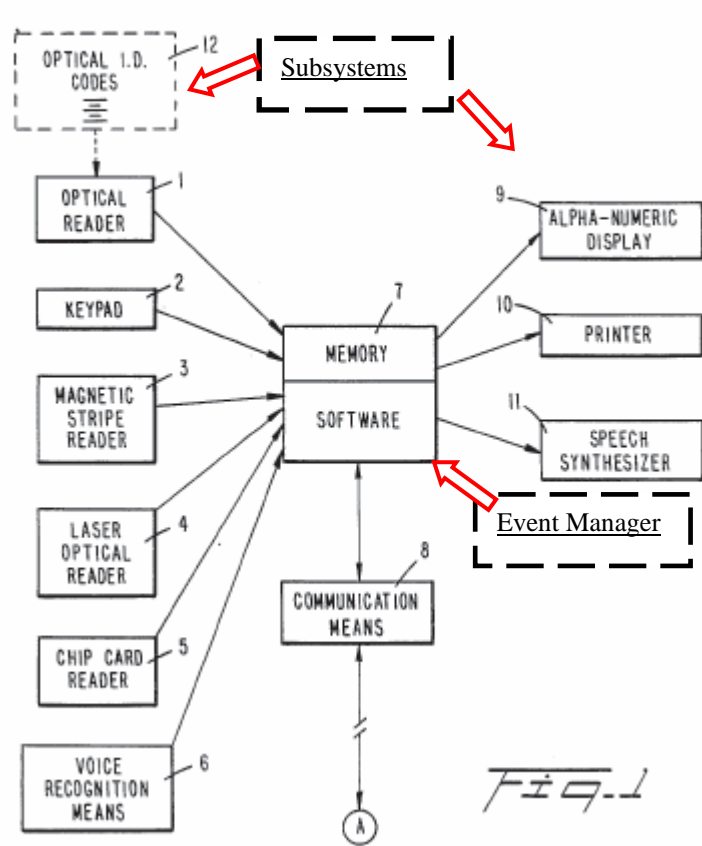
	<p>mailbox.”</p> <p>Col 10 Lines 21-24:</p> <p>“...the manufacturer host...during its polling of requests and other items placed in its mailbox, senses that an order has been placed...”</p>
<p>inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and</p>	<p>Col. 9 Lines 14-16:</p> <p>“The manufacturer host can then decode each item on the quote and price each item.”</p> <p>Col. 10 Lines 24-26:</p> <p>“The manufacturer host proceeds to verify the accuracy of the order...”</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	<p>Col. 9 Lines 17-24:</p> <p>“The manufacturer host can reassembles the file as a price quoted for transmittal... Again the quote is transmitted into the electronic mail system.”</p> <p>Col 10 Lines 27-29:</p> <p>“The manufacturer host...prints the order for scheduling and credit approval”</p>

U.S. Pat. No 6,067,525 claim 20	Long et al U.S. Pat. No. 5,117,354
<p>20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:</p>	<p>The preamble is not a limitation, nonetheless col. 1 lines 7-13:</p> <p>“... present invention relates to systems for pricing and ordering goods... so that sales representatives can obtain pricing information, and place orders for the goods to be manufactured...”</p>  <p>Col. 3 lines 27-34:</p> <p>A variety of sales representatives in the field are each equipped with a personal computer 12 having processing capabilities, local memory and long term storage such as disk drives. Those personal computers 12 may be installed at the office locations of the sales representatives or may be portable units which they may carry with them to their home or other remote locations.</p> <p>Col. 2 lines 38-45:</p> <p>“...a central data processing facility connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative... an electronic mail serving host,... the manufacturing host...”</p>
<p>automatically detecting one or more changes in state characteristic of an event occurring in the sales process;</p>	<p>Co. 9 Lines 5-10:</p> <p>“...software embedded in step 80 [host] ...looks for mail placed in its mailbox.”</p> <p>Col 10 Lines 21-24:</p> <p>“...the manufacturer host...during its polling of requests and other items placed in its mailbox, senses that an order has been placed...”</p>
<p>inferring</p>	<p>Col. 9 Lines 14-16:</p>

<p>occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state; and</p>	<p>“The manufacturer host can then decode each item on the quote and price each item.”</p> <p>Col. 10 Lines 24-26:</p> <p>“The manufacturer host proceeds to verify the accuracy of the order...”</p>
<p>automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.</p>	<p>Col. 9 Lines 17-24:</p> <p>“The manufacturer host can reassembles the file as a price quoted for transmittal... Again the quote is transmitted into the electronic mail system.”</p> <p>Col 10 Lines 27-29:</p> <p>“The manufacturer host...prints the order for scheduling and credit approval”</p>

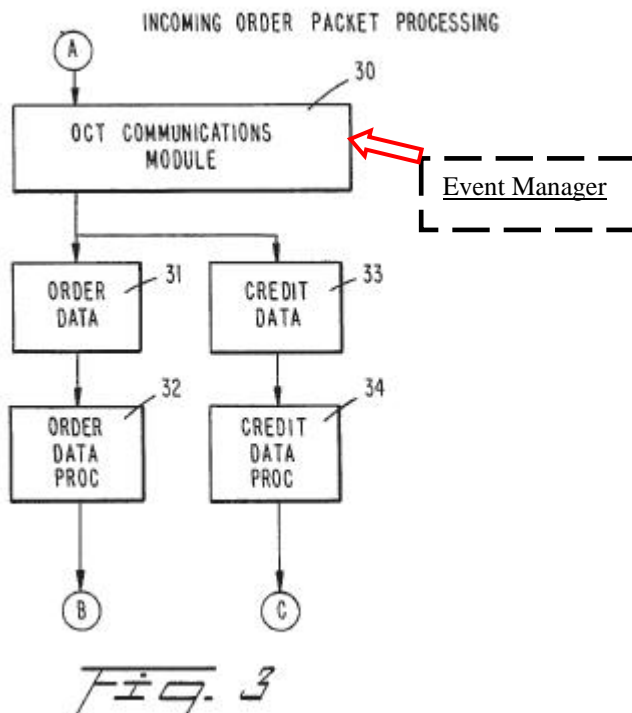
U.S. Pat. No 6,067, 525 claim 40	Long et al U.S. Pat. No. 5,117,354
40. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation, nonetheless col. 1 lines 7-13:</p> <p>“... present invention relates to systems for pricing and ordering goods... so that sales representatives can obtain pricing information, and place orders for the goods to be manufactured...”</p>
<p>a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and</p>	 <p>Col. 3 lines 27-34:</p> <p>A variety of sales representatives in the field are each equipped with a personal computer 12 having processing capabilities, local memory and long term storage such as disk drives. Those personal computers 12 may be installed at the office locations of the sales representatives or may be portable units which they may carry with them to their home or other remote locations.</p> <p>Col. 2 lines 38-45:</p> <p>“...a central data processing facility connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative... an electronic mail serving host,... the manufacturing host...”</p>
<p>an event manager coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring in</p>	<p>Col. 2 lines 38-45:</p> <p>“...a central data processing facility connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative... an electronic mail serving host,... the manufacturing host...”</p> <p>Co. 9 Lines 5-10:</p> <p>“...software embedded in step 80 [host] ...looks for mail placed in its mailbox.”</p>

the system,	Col 10 Lines 21-24: “...the manufacturer host...during its polling of requests and other items placed in its mailbox, senses that an order has been placed...”
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state,	Col. 9 Lines 14-16: “The manufacturer host can then decode each item on the quote and price each item.” Col. 10 Lines 24-26: “The manufacturer host proceeds to verify the accuracy of the order...”
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	Col. 9 Lines 17-24: “The manufacturer host can reassembles the file as a price quoted for transmittal... Again the quote is transmitted into the electronic mail system.” Col 10 Lines 27-29: “The manufacturer host...prints the order for scheduling and credit approval”
automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	Col. 9 Lines 17-24: “The manufacturer host can reassembles the file as a price quoted for transmittal... Again the quote is transmitted into the electronic mail system.” Col 10 Lines 27-29: “The manufacturer host...prints the order for scheduling and credit approval”

U.S. Pat. No 6,067,525 claim 1	Gorog U.S. Pat. No. 4,947,028
1. A computer implemented sales system used to facilitate a sales process, the system comprising:	<p>The preamble is not a limitation, nonetheless col. 1 lines 10-13:</p> <p>“The originality of the invention lies in the integration of existing devices, products, and networks to accomplish a unique service which will making the process of buying and selling significantly more efficient.”</p> <p>Col. 7 lines 39-43:</p> <p>“In summary, this process selects the merchant/supplier, confirms the availability of inventory to fulfill the sale, confirms the price, method of payment, and credit status of the consumer as well as the delivery date and method of delivery.”</p>
a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and	<p style="text-align: center;">ORDER COMPUTER TERMINAL "OCT"</p>  <p>The diagram illustrates the architecture of the ORDER COMPUTER TERMINAL "OCT". At the center is a block containing "MEMORY" (7) and "SOFTWARE" (8). To the left, several input devices are connected to the central block: "OPTICAL I.D. CODES" (12) via "OPTICAL READER" (1), "KEYPAD" (2), "MAGNETIC STRIPE READER" (3), "LASER OPTICAL READER" (4), "CHIP CARD READER" (5), and "VOICE RECOGNITION MEANS" (6). To the right, output devices are connected: "ALPHA-NUMERIC DISPLAY" (9), "PRINTER" (10), and "SPEECH SYNTHESIZER" (11). Below the central block is "COMMUNICATION MEANS" (8), which is connected to a power source (A). Two dashed boxes with red arrows highlight specific components: "Subsystems" (enclosing the input devices) and "Event Manager" (enclosing the output devices and communication means). The diagram is labeled "Fig. 1" at the bottom right.</p> <p>Col. 2 lines 54-68:</p> <p>“(c) A order computer terminal ("OCT") with means to input data orally, optically, magnetically, electronically, and manually having associated order processing software and communications capabilities allowing receipt of communications from the CCS and further providing output communications to the CCS.</p> <p>The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other</p>

	<p>computers using accepted industry protocols.”</p> <p>Col. 9 lines 8-30:</p> <p>“An automated order and payment [sales] system, which comprises: ... A central data processing means with communication capability adapted to receive information from a plurality of remote programmable data input/output means...”</p>
<p>an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system</p>	<p>Col. 6 lines 17-29:</p> <p>Referring to FIG. 3, the CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.</p> <p>Col. 2 lines 46-50:</p> <p>“(a) A central computer system ("CCS") with a variety of programs, processing and storage capability and communications capabilities to allow input and output communications with order computer terminals.”</p> <p>Col. 2 lines 61-68:</p> <p>“The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.”</p> <p>Col. 3 lines 1-8:</p> <p>“The CCS has various computer software programs that allow product/service order information to be accepted and transmitted from the central computer. Such software will also confirm or deny orders for products based upon records of inventories that have been provided by participating businesses or by sending a query to other computers holding the necessary data records for participating businesses.”</p> <p>Col. 9 lines 23-26:</p> <p>“A central data processing [event manager] means with communication capability [coupled to] adapted to receive information from a plurality of remote programmable data input/output means [subsystems] ...”</p> <p>Reexamination col. 1 Line 68 – col. 2 lines 1-4:</p> <p>“said central data processor [event manager] comprising:... means for receiving [detecting changes in state characteristic of an event] first [from optical reader] and second [from payment card reader] data...”</p>

inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and



Col. 6 lines 37-51:

“[T]he CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.”

Reexamination col. 2 lines 1-17:

“means for receiving... first and second data...order confirmation [inferring an event] means ... subsequent to receipt of payment authorization from the external database...”

automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.

Reexamination col. 2 lines 18-23:

“means for transmitting at least the first data and the payment authorization information to a product/service provider in accordance with the first data, in response to receipt by the central data processor of an order confirmation message from the remote terminal”

**U.S. Pat. No
6,067,525 claim 20**

20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:

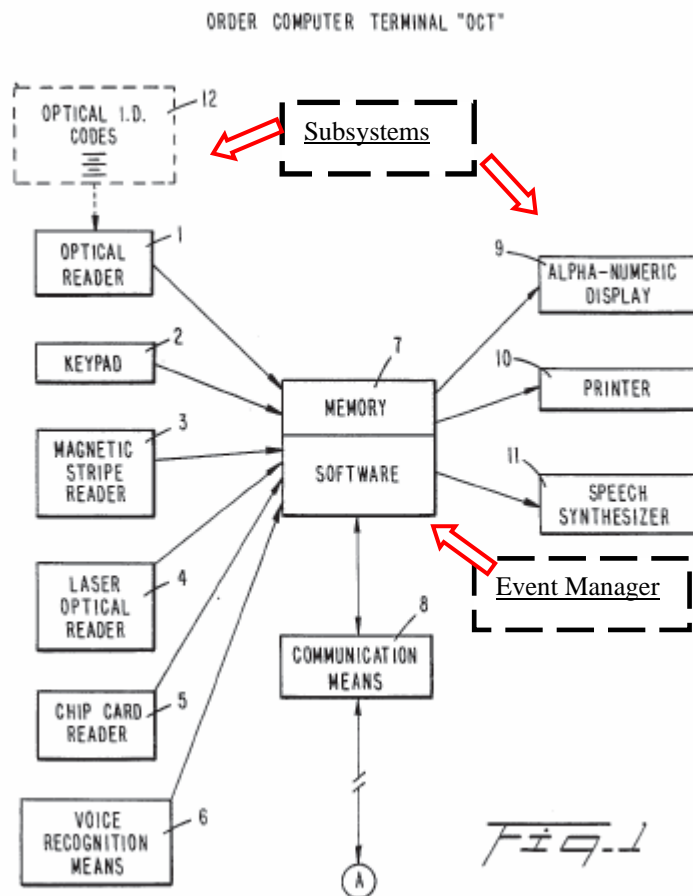
Gorog U.S. Pat. No. 4,947,028

The preamble is not a limitation, nonetheless col. 1 lines 10-13:

“The originality of the invention lies in the integration of existing devices, products, and networks to accomplish a unique service which will making the process of buying and selling significantly more efficient.”

Col. 7 lines 39-43:

“In summary, this process selects the merchant/supplier, confirms the availability of inventory to fulfill the sale, confirms the price, method of payment, and credit status of the consumer as well as the delivery date and method of delivery.”



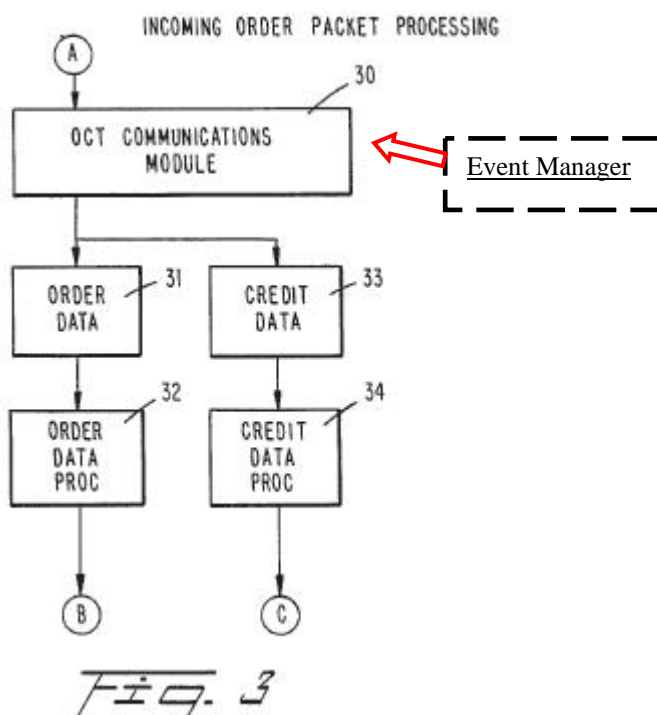
Col. 2 lines 54-68:

“(c) A order computer terminal ("OCT") with means to input data orally, optically, magnetically, electronically, and manually having associated order processing software and communications capabilities allowing receipt of communications from the CCS and further providing output communications to the CCS.

The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or

	<p>other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.”</p> <p>Col. 9 lines 8-30:</p> <p>“An automated order and payment [sales] system, which comprises: ... A central data processing means with communication capability adapted to receive information from a plurality of remote programmable data input/output means...”</p>
<p>automatically detecting one or more changes in state characteristic of an event occurring in the sales process;</p>	<p>Col. 6 lines 17-29:</p> <p>Referring to FIG. 3, the CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.</p> <p>Col. 2 lines 46-50:</p> <p>“(a) A central computer system ("CCS") with a variety of programs, processing and storage capability and communications capabilities to allow input and output communications with order computer terminals.”</p> <p>Col. 2 lines 61-68:</p> <p>“The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.”</p> <p>Col. 3 lines 1-8:</p> <p>“The CCS has various computer software programs that allow product/service order information to be accepted and transmitted from the central computer. Such software will also confirm or deny orders for products based upon records of inventories that have been provided by participating businesses or by sending a query to other computers holding the necessary data records for participating businesses.”</p> <p>Reexamination col. 1 Line 68 – col. 2 lines 1-4:</p> <p>“said central data processor [event manager] comprising:... means for receiving [detecting changes in state characteristic of an event] first [from optical reader] and second [from payment card reader] data...”</p>

inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state; and



Col. 6 lines 37-51:

“[T]he CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.”

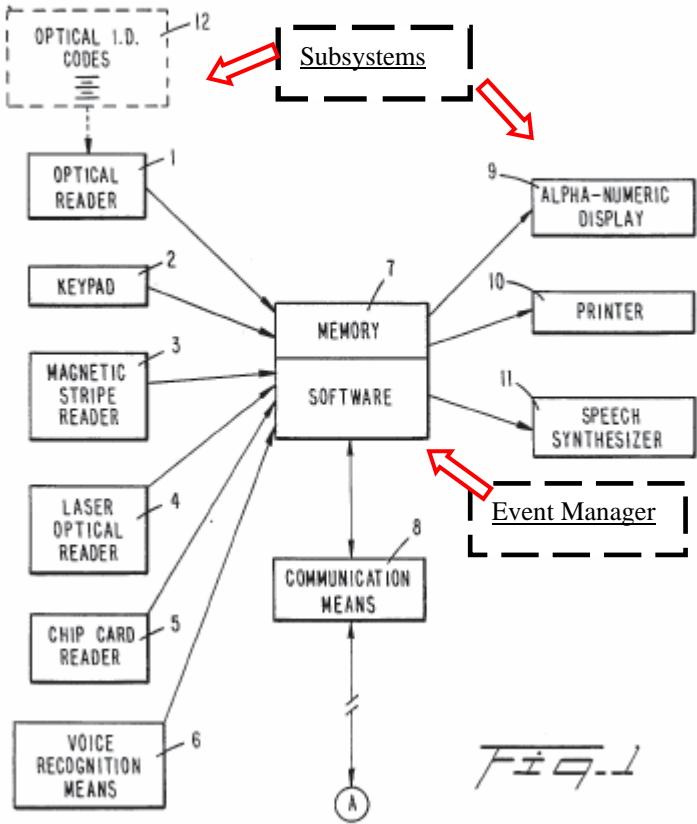
Reexamination col. 2 lines 1-17:

“means for receiving... first and second data...order confirmation [inferring an event] means ... subsequent to receipt of payment authorization from the external database...”

automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.

Reexamination col. 2 lines 18-23:

“means for transmitting at least the first data and the payment authorization information to a product/service provider in accordance with the first data, in response to receipt by the central data processor of an order confirmation message from the remote terminal”

U.S. Pat. No. 6,067, 525 claim 40	Gorog U.S. Pat. No. 4,947,028
<p>40. A computer implemented sales system used to facilitate a sales process, the system comprising:</p>	<p>The preamble is not a limitation, nonetheless col. 1 lines 10-13: “The originality of the invention lies in the integration of existing devices, products, and networks to accomplish a unique service which will making the process of buying and selling significantly more efficient.” Col. 7 lines 39-43: “In summary, this process selects the merchant/supplier, confirms the availability of inventory to fulfill the sale, confirms the price, method of payment, and credit status of the consumer as well as the delivery date and method of delivery.”</p>
<p>a plurality of subsystems configured to electronically facilitate actions performed during the sales process; and</p>	<p>ORDER COMPUTER TERMINAL "OCT"</p>  <p>The diagram, labeled FIG. 1, illustrates the architecture of the ORDER COMPUTER TERMINAL "OCT". At the top, a dashed box labeled "Subsystems" contains an "OPTICAL I.D. CODES" block (12). Below this, a vertical column of input devices is connected to a central block containing "MEMORY" (7) and "SOFTWARE" (8). These input devices include an "OPTICAL READER" (1), "KEYPAD" (2), "MAGNETIC STRIPE READER" (3), "LASER OPTICAL READER" (4), "CHIP CARD READER" (5), and "VOICE RECOGNITION MEANS" (6). The central "MEMORY" and "SOFTWARE" block is also connected to a "COMMUNICATION MEANS" block (8) at the bottom, which is linked to a power source "A". To the right of the central block, three output devices are shown: an "ALPHA-NUMERIC DISPLAY" (9), a "PRINTER" (10), and a "SPEECH SYNTHESIZER" (11). A dashed box labeled "Event Manager" is positioned between the central block and the output devices, with red arrows indicating bidirectional communication between the central block and the "Event Manager", and between the "Event Manager" and the output devices. The label "FIG. 1" is handwritten at the bottom right of the diagram.</p> <p>Col. 2 lines 54-68: “(c) A order computer terminal ("OCT") with means to input data orally, optically, magnetically, electronically, and manually having associated order processing software and communications capabilities allowing receipt of communications from the CCS and further providing output communications to the CCS. The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such</p>

	<p>terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.”</p> <p>Col. 9 lines 8-30:</p> <p>“An automated order and payment [sales] system, which comprises: ... A central data processing means with communication capability adapted to receive information from a plurality of remote programmable data input/output means...”</p>
<p>an event manager coupled to the subsystems and configured to detect one or more changes in state characteristic of an event occurring in the system,</p>	<p>Col. 6 lines 17-29:</p> <p>Referring to FIG. 3, the CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.</p> <p>Col. 2 lines 46-50:</p> <p>“(a) A central computer system ("CCS") with a variety of programs, processing and storage capability and communications capabilities to allow input and output communications with order computer terminals.”</p> <p>Col. 2 lines 61-68:</p> <p>“The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.”</p> <p>Col. 3 lines 1-8:</p> <p>“The CCS has various computer software programs that allow product/service order information to be accepted and transmitted from the central computer. Such software will also confirm or deny orders for products based upon records of inventories that have been provided by participating businesses or by sending a query to other computers holding the necessary data records for participating businesses.”</p> <p>Col. 9 lines 23-26:</p> <p>“A central data processing [event manager] means with communication capability [coupled to] adapted to receive information from a plurality of remote programmable data input/output means [subsystems] ...”</p> <p>Reexamination col. 1 Line 68 – col. 2 lines 1-4:</p> <p>“said central data processor [event manager] comprising:... means for receiving</p>

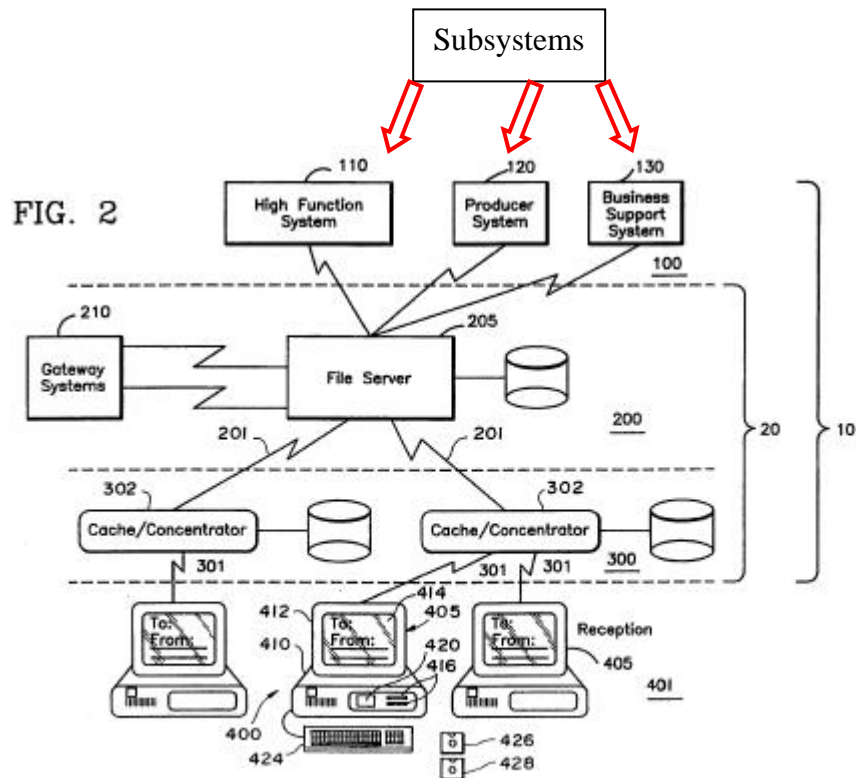
	[detecting changes in state characteristic of an event] first [from optical reader] and second [from payment card reader] data...”
infer occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state,	<p style="text-align: center;"><i>Fig. 3</i></p> <p>Col. 6 lines 37-51:</p> <p>“[T]he CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.”</p> <p>Reexamination col. 2 lines 1-17:</p> <p>“means for receiving... first and second data...order confirmation [inferring an event] means ... subsequent to receipt of payment authorization from the external database...”</p>
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	<p>Reexamination col. 2 lines 18-23:</p> <p>“means for transmitting at least the first data and the payment authorization information to a product/service provider in accordance with the first data, in response to receipt by the central data processor of an order confirmation message from the remote terminal”</p>

automatically initiate an operation using one or more of the plurality of subsystems to facilitate the action to be performed based on the inferred context.	Reexamination col. 2 lines 18-23: “means for transmitting at least the first data and the payment authorization information to a product/service provider in accordance with the first data, in response to receipt by the central data processor of an order confirmation message from the remote terminal”
--	---

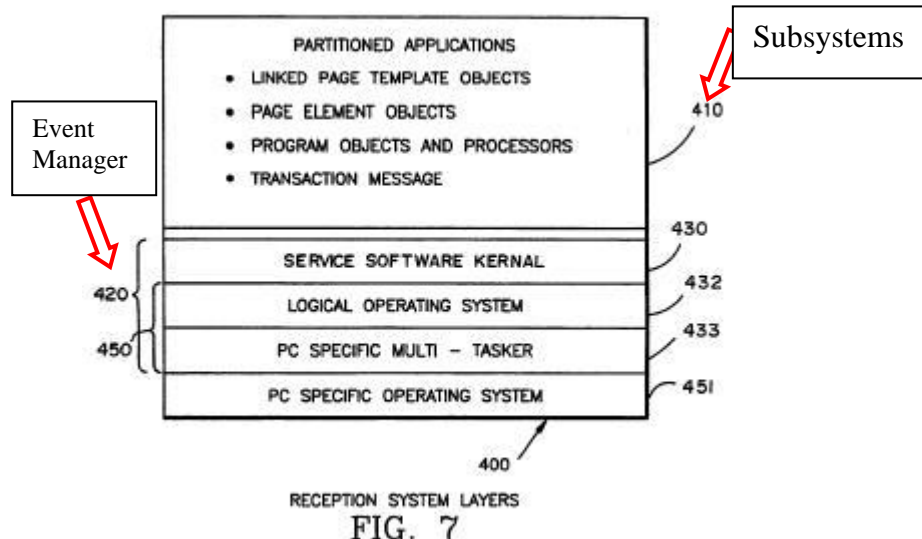
U.S. Pat. No 6,067,525 claim 1	Filepp et al U.S. Pat. No. 5,347,632
<p>1. A computer implemented sales system used to facilitate a sales process, the system comprising:</p>	<p>The preamble is not a limitation, nonetheless abstract:</p> <p>“An interactive computer system that enables a user to... perform desired transactions such as banking and shopping...”</p> <p>Col. 6 Lines 56-61:</p> <p>“Services available to the user include...the purchase of items such as retail merchandise and groceries... and buy/sell orders for stocks and bonds.”</p>
<p>a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and</p>	<div data-bbox="560 609 1474 1144" data-label="Diagram"> <p>FIG. 7</p> </div> <p>Col. 3 lines 27-34:</p> <p>“In preferred form the reception system further comprises...a plurality of partitioned applications; and object processing means...for selecting and retrieving objects...and interpreting and executing the partitioned applications”</p> <p>Col. 5 lines 26-27:</p> <p>“Each application partition is an independent, self-contained unit and can operate correctly by itself.”</p> <p>Col. 6 lines 45-68:</p> <p>“Services available to the user include display of information such as movie reviews, the latest news, airlines reservations, the purchase of items such as retail merchandise and groceries, and quotes and buy/sell orders for stocks and bonds. Network 10 provides an environment in which a user, via RS 400 establishes a session with the network and accesses a large number of services. These services are specifically constructed applications which as noted are partitioned so they may be distributed without undo transmission time, and may be processed and selectively stored on a user's RS 400 unit.”</p>

Col. 9, lines 30-34:

“Advertisements 280 may be presented to the user on an individual basis from queues of advertisements that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400.”



an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system



Col. 82 lines 30-59:

“Again with reference to FIG. 7, native software 420 ... is composed of two components: the service software 430 and the operating environment 450. ... Service software 430 provides functions specific

to providing interaction between the user and interactive network 10 ...

Service software 430 is comprised of modules, which are device-independent software components that together obtain, interpret and store partitioned applications existing as a collection of objects. The functions performed by, and the relationship between, the service software 430 module is shown in FIG. 8 and discussed further below.”

Col. 83 lines 12-21:

“RS native software provides a virtual machine interface for partitioned applications, such that all objects comprising partitioned applications "see" the same machine. RS native software provides support for the following functions: (1) keyboard and mouse input; (2) text and graphics display; (3) application interpretation; (4) application database management; (5) local application storage; (6) network and link level communications; (7) user activity data collection; and (8) advertisement management.”

Col. 8 lines 9-14:

“The RS 400 is the point of application session control because it has the ability to select and randomly access objects representing all or part of partitioned applications and their data. RS 400 processes objects according to information contained therein and events created by the user on personal computer 405.”

Col. 3 lines 27-34:

“In preferred form the reception system further comprises... objects comprising a plurality of partitioned applications; and object processing means...for selecting and retrieving objects...and interpreting and executing the partitioned applications”

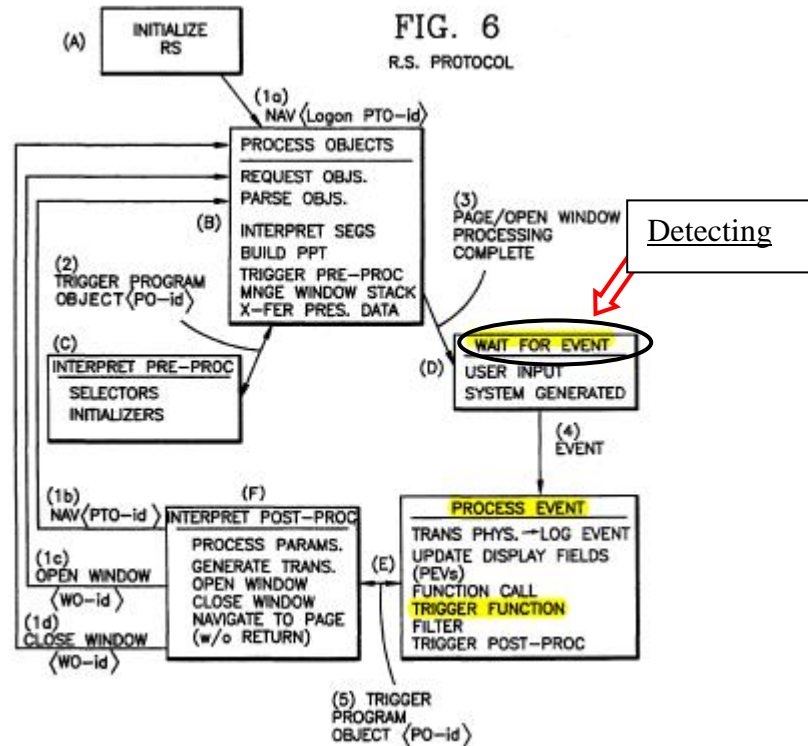
Col. 6 lines 3-9:

“The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the partitioned application.”

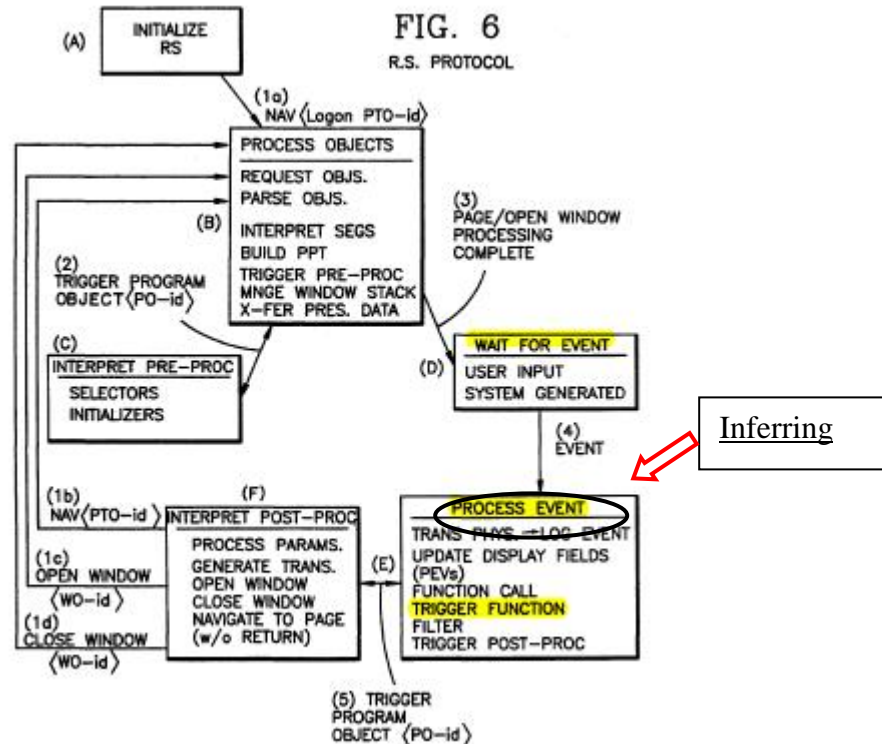
Col. 101 lines 53-54:

“receiving requests for partitioned applications at the reception

system.”



inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and



Col. 14 lines 52-55:

“Program events will be specified in logical terms and will be mapped by the reception system to specific physical triggers...”

Col. 81 lines 43-51:

“Certain inputs, such as RETURN or mouse clicks in particular fields, are mapped to logical events by keyboard manager 434, which are called completion (or commit) events. Completion events signify the completion of some selection or specification process associated with the partitioned application and trigger a partition level and/or page level post-processor to process the ‘action’ parameters associated with the user's selection and commit event.”

Col. 39 lines 60-66:

“Reception system is aware of the occurrence of physical events during the...interactive sessions. When a physical event such as the depression of a ...key corresponds to a logical event such as the completion of data entry in a field...”

Col. 73 lines 52-64:

“Through this interaction, the user is able to input data into fields provided as part of the display, or may individually select choices causing a standard or personalized page to be built (as explained below) for display on the monitor of personal computer 405. Such inputs will cause RS 400 [reception system] to interpret events and trigger pre-processors or post-processors, retrieve specified objects, communicate with system components, control user options, cause the display of advertisements on a page, open or close window partitions to provide additional navigation possibilities, and collect and report data about events, including certain types of objects processed.”

Col. 5, line 55 – Col. 6 line 9:

“Objects carry application programs and information for display at monitor screen 414 of RS 400. Application program objects, called pre-processor and post-processors, set up the environment for the user's interaction with network 10 and respond to events created when the user inputs information at keyboard 424 of RS 400. Such events typically trigger a program object to be processed, causing one of the following: sending of transactional information to the coapplications in one layer of the network 10; the receiving of information for use in programs or for presentation in application-dependent fields on monitor screen 414; or the requesting of a new objects to be processed by RS 400. Such objects may be part of the same application or a completely new application. The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the partitioned application.

Col. 6 lines 10-12:

RS 400 includes a means to communicate with network 10 to retrieve objects in response to events occurring at RS 400 and to send and receive messages.

Col. 7 lines 35-46:

Objects may contain: control information; program instruction to set up an application processing environment and to process user or network created events; information about what is to be displayed and how it is to be displayed; references to programs to be interpretively executed; and references to other objects, which may be called based upon certain conditions or the occurrence of certain events at the user's personal computer, resulting in the selection and retrieval of other partitioned applications packaged as objects.

Col. 8 lines 2-8:

If such objects are requested by the RS 400, the cache/concentrator 302 automatically requests the object from file server 205. The requested object is routed back to the requesting cache/concentrator 302, which automatically routes it to the communications line on which the request was originally made, from which it is received by the RS 400.

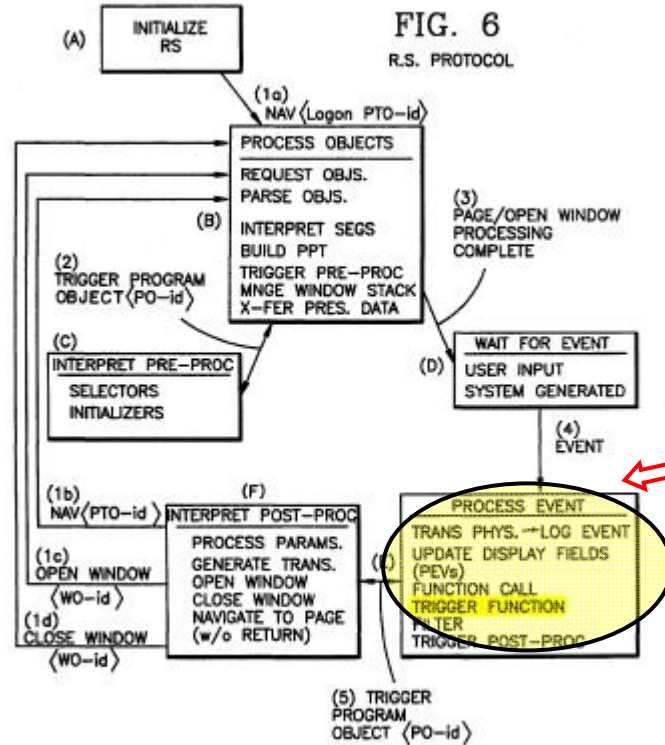
Col. 9, lines 35-47:

Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

Col. 81 lines 15-22:

“This feature enables RS 400 to conditionally deliver information to the user base upon predetermined parameters, such as his personal demographics or locale. For example, the parameters specified may be the transaction codes required to retrieve the user's age, sex, and personal interest codes from records contained in user profiles stored at the switch/file server layer 200.”

automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.



Automatically
initiating

Col. 74 lines 59-62:

“If the functions are part of [reception system] they can be altered or extended... [to] permit the execution of program objects to be triggered...”

Col. 8 lines 24-27:

“selecting another partitioned application to be processed upon a user generated completion event for the current partitioned application.”

Col. 99 lines 12-16:

The reception system comprising: “object processing means responsive to the input means for selectively retrieving and interpreting objects to extract data and program instructions for composing and generating the partitioned applications...”

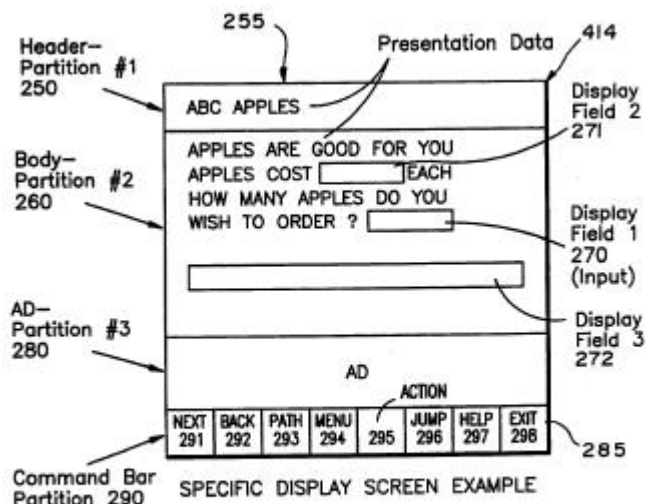


FIG. 3b

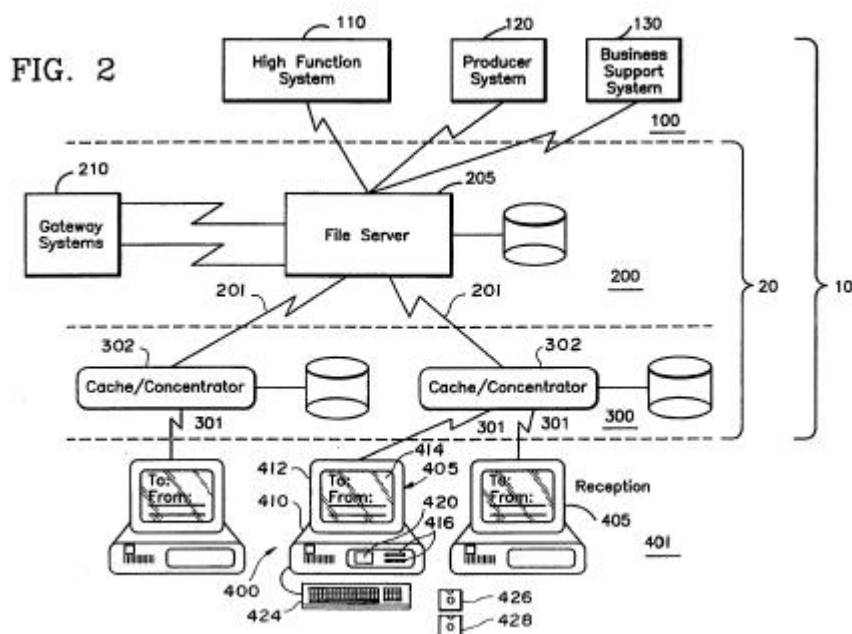
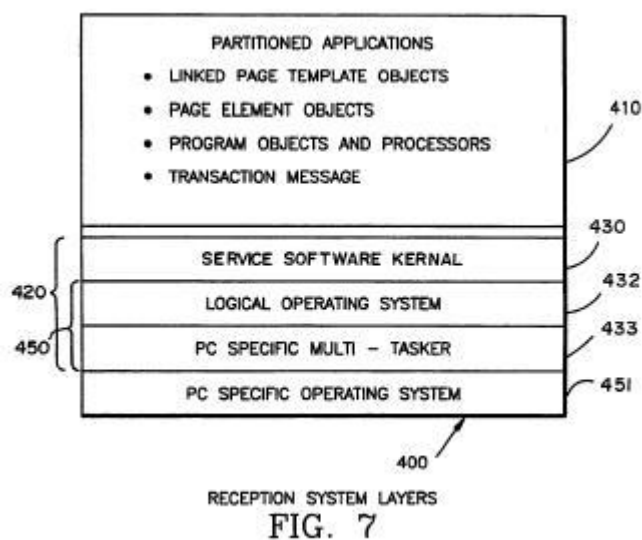
Col. 95 line 67- Col. 96 line 3:

“The page illustrated in FIG. 3(b) corresponds to a partitioned application that permit's [sic] a personal computer user to purchase apples. It shows how the monitor screen 414 of personal computer 405 might appear to the user.”

Col. 9, lines 35-47:

Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

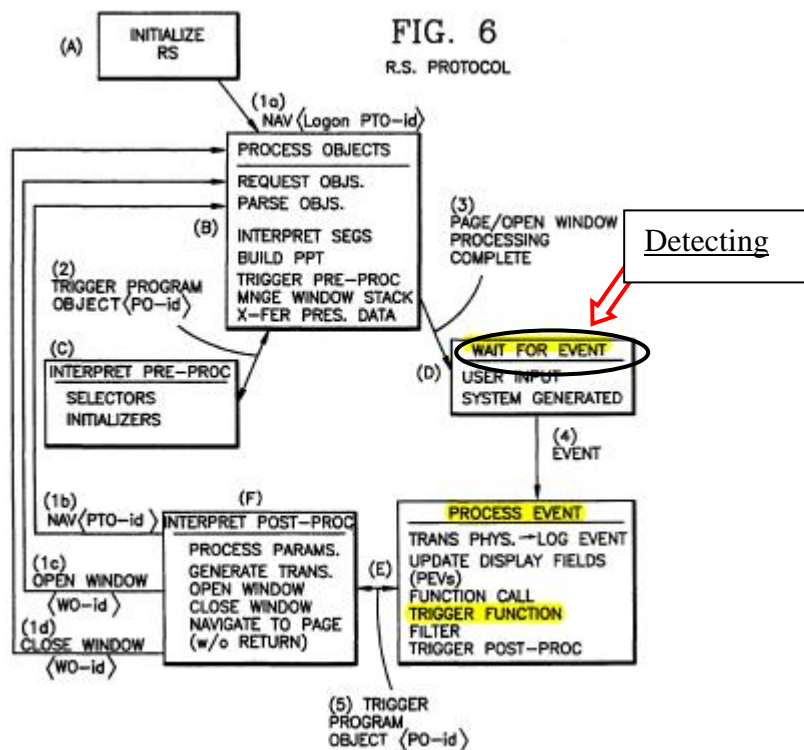
U.S. Pat. No 6,067,525 claim 20	Filepp et al U.S. Pat. No. 5,347,632
<p>20. A method of facilitating a sales process using a computer arrangement having a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process, the method comprising the steps of:</p>	<p>The preamble is not a limitation, nonetheless abstract:</p> <p>“An interactive computer system that enables a user to... perform desired transactions such as banking and shopping...”</p> <p>Col. 6 lines 56-61:</p> <p>“Services available to the user include...the purchase of items such as retail merchandise and groceries... and buy/sell orders for stocks and bonds.”</p> <p>Col. 3 lines 27-34:</p> <p>“In preferred form the reception system further comprises...a plurality of partitioned applications; and object processing means...for selecting and retrieving objects...and interpreting and executing the partitioned applications”</p> <p>Col. 5 lines 26-27:</p> <p>“Each application partition is an independent, self-contained unit and can operate correctly by itself.”</p> <p>Col. 6 lines 45-68:</p> <p>“Services available to the user include display of information such as movie reviews, the latest news, airlines reservations, the purchase of items such as retail merchandise and groceries, and quotes and buy/sell orders for stocks and bonds. Network 10 provides an environment in which a user, via RS 400 establishes a session with the network and accesses a large number of services. These services are specifically constructed applications which as noted are partitioned so they may be distributed without undo transmission time, and may be processed and selectively stored on a user's RS 400 unit.”</p> <p>Col. 9, lines 30-34:</p> <p>“Advertisements 280 may be presented to the user on an individual basis from queues of advertisements that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400.”</p>



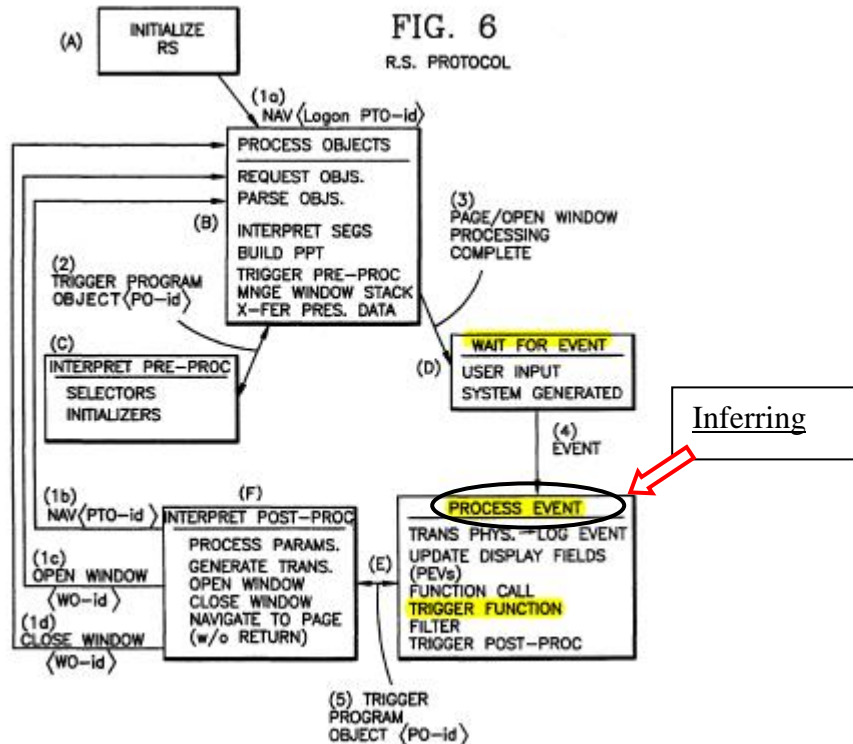
automatically detecting one or more changes in state characteristic of an event occurring in the sales process;

Col. 101 lines 53-54:

“receiving requests for partitioned applications at the reception system.”



inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state; and



Col. 14 lines 52-55:

“Program events will be specified in logical terms and will be mapped by the reception system to specific physical triggers...”

Col. 81 lines 43-51:

“Certain inputs, such as RETURN or mouse clicks in particular fields, are mapped to logical events by keyboard manager 434, which are called completion (or commit) events. Completion events signify the completion of some selection or specification process associated with the partitioned application and trigger a partition level and/or page level post-processor to process the ‘action’ parameters associated with the user's selection and commit event.”

Col. 39 lines 60-66:

“Reception system is aware of the occurrence of physical events during the...interactive sessions. When a physical event such as the depression of a ...key corresponds to a logical event such as the completion of data entry in a field...”

Col. 73 lines 52-64:

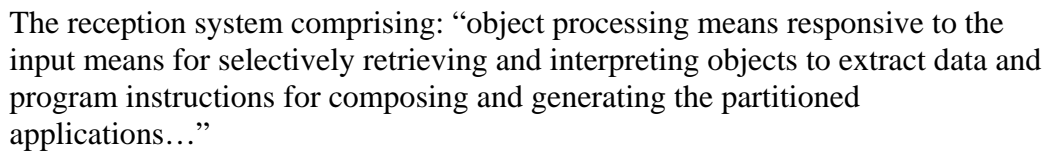
“Through this interaction, the user is able to input data into fields provided as part of the display, or may individually select choices causing a standard or personalized page to be built (as explained below) for display on the monitor of personal computer 405. Such inputs will cause RS 400 [reception system] to interpret events and trigger pre-processors or post-processors, retrieve specified objects, communicate with system components, control user options, cause the display of advertisements on a page, open or close window partitions to provide additional navigation possibilities, and collect and report data about events, including certain types of objects processed.”

Col. 9, lines 35-47:

Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

Col. 81 lines 15-22:

“This feature enables RS 400 to conditionally deliver information to the user base upon predetermined parameters, such as his personal demographics or locale. For example, the parameters specified may be the transaction codes required to retrieve the user's age, sex, and personal interest codes from records contained in user profiles stored at the switch/file server layer 200.”



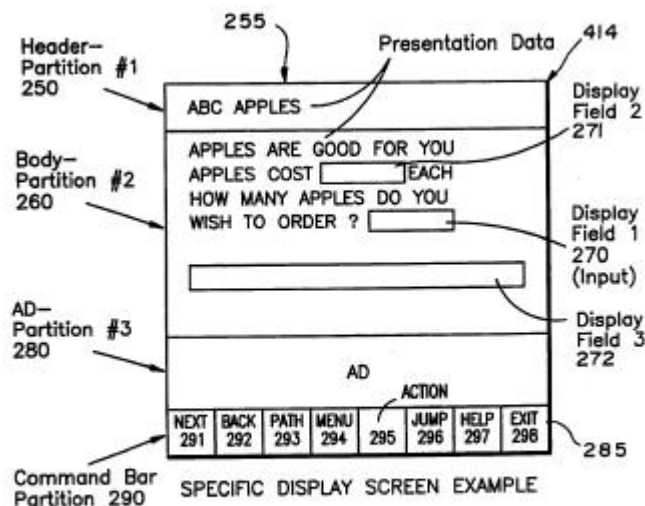


FIG. 3b

Col. 95 line 67- Col. 96 line 3:

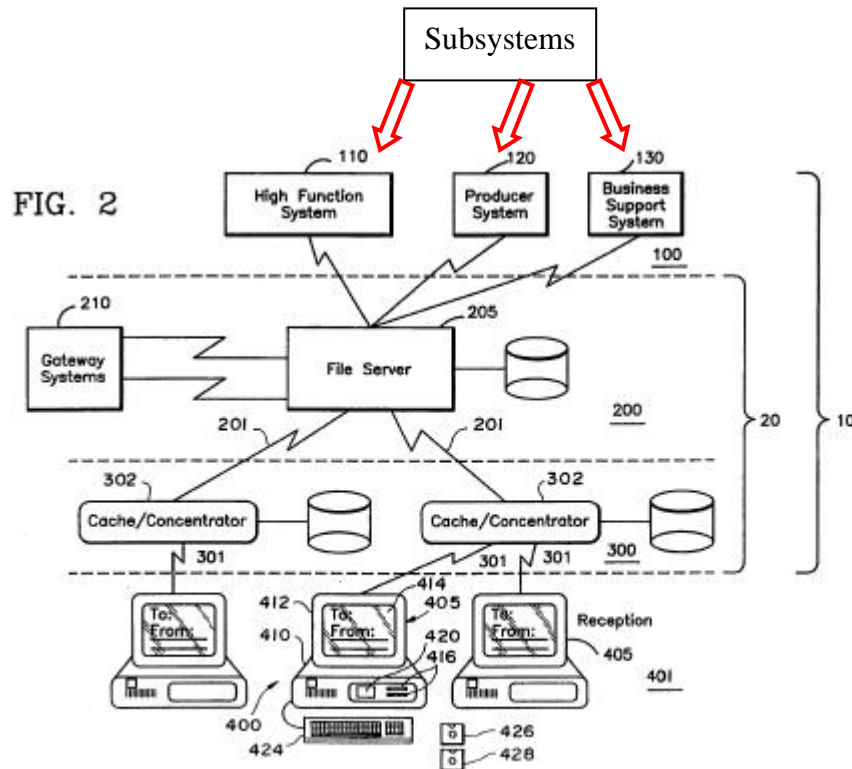
“The page illustrated in FIG. 3(b) corresponds to a partitioned application that permit's [sic] a personal computer user to purchase apples. It shows how the monitor screen 414 of personal computer 405 might appear to the user.”

Col. 9, lines 35-47:

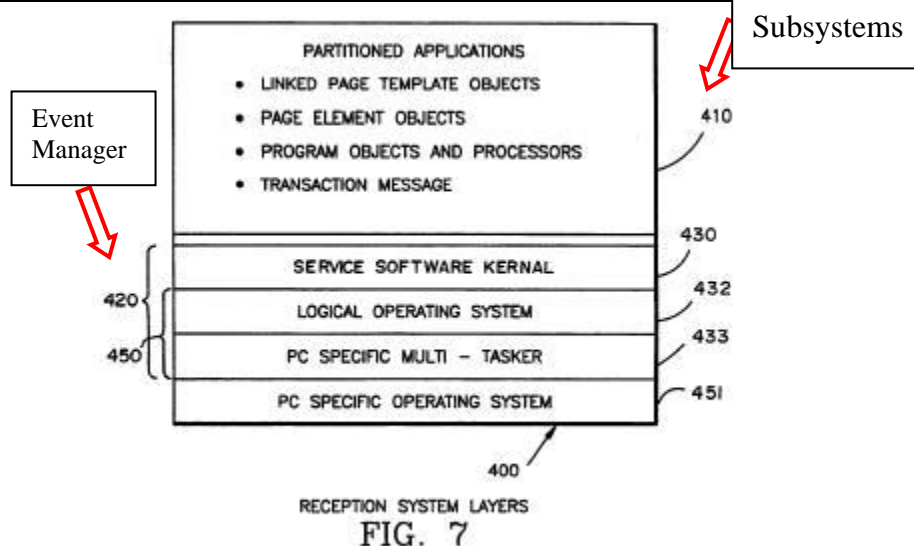
Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

<p>U.S. Pat. No 6,067, 525 claim 40</p>	<p>Filepp et al U.S. Pat. No. 5,347,632</p>
<p>40. A computer implemented sales system used to facilitate a sales process, the system comprising:</p>	<p>The preamble is not a limitation, nonetheless abstract: “An interactive computer system that enables a user to... perform desired transactions such as banking and shopping...” Col. 6 Lines 56-61: “Services available to the user include...the purchase of items such as retail merchandise and groceries... and buy/sell orders for stocks and bonds.”</p>
<p>a plurality of subsystems configured to facilitate one or more actions performed during at least one phase of the sales process; and</p>	<div data-bbox="354 611 1260 1167" data-label="Diagram"> </div> <p>Col. 3 lines 27-34: “In preferred form the reception system further comprises...a plurality of partitioned applications; and object processing means...for selecting and retrieving objects...and interpreting and executing the partitioned applications” Col. 5 lines 26-27: “Each application partition is an independent, self-contained unit and can operate correctly by itself.” Col. 6 lines 45-68: “Services available to the user include display of information such as movie reviews, the latest news, airlines reservations, the purchase of items such as retail merchandise and groceries, and quotes and buy/sell orders for stocks and bonds. Network 10 provides an environment in which a user, via RS 400 establishes a session with the network and accesses a large number of services. These services are specifically constructed applications which as noted are partitioned so they may be distributed without undo transmission time, and may be processed and selectively stored on a user's RS 400 unit.” Col. 9, lines 30-34:</p>

“Advertisements 280 may be presented to the user on an individual basis from queues of advertisements that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400.”



an event manager, coupled to the subsystems, the event manager detecting one or more changes in state characteristic of an event occurring within the system



Col. 82 lines 30-59:

“Again with reference to FIG. 7, native software 420 ... is composed of two components: the service software 430 and the operating environment 450. ... Service software 430 provides functions specific to providing interaction between the user and

interactive network 10 ...

Service software 430 is comprised of modules, which are device-independent software components that together obtain, interpret and store partitioned applications existing as a collection of objects. The functions performed by, and the relationship between, the service software 430 module is shown in FIG. 8 and discussed further below.”

Col. 83 lines 12-21:

“RS native software provides a virtual machine interface for partitioned applications, such that all objects comprising partitioned applications "see" the same machine. RS native software provides support for the following functions: (1) keyboard and mouse input; (2) text and graphics display; (3) application interpretation; (4) application database management; (5) local application storage; (6) network and link level communications; (7) user activity data collection; and (8) advertisement management.”

Col. 8 lines 9-14:

“The RS 400 is the point of application session control because it has the ability to select and randomly access objects representing all or part of partitioned applications and their data. RS 400 processes objects according to information contained therein and events created by the user on personal computer 405.”

Col. 3 lines 27-34:

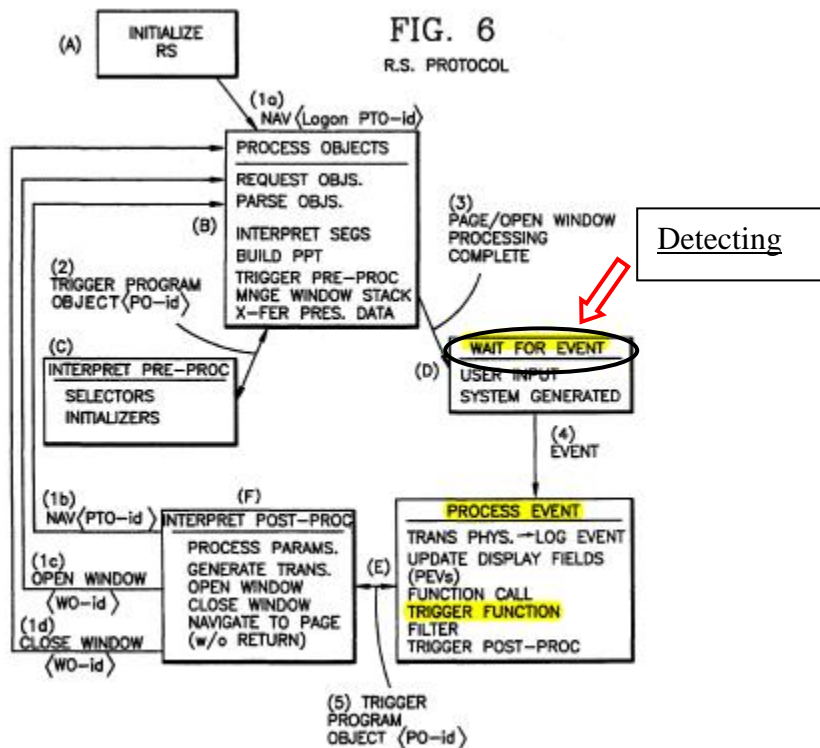
“In preferred form the reception system further comprises... objects comprising a plurality of partitioned applications; and object processing means...for selecting and retrieving objects...and interpreting and executing the partitioned applications”

Col. 6 lines 3-9:

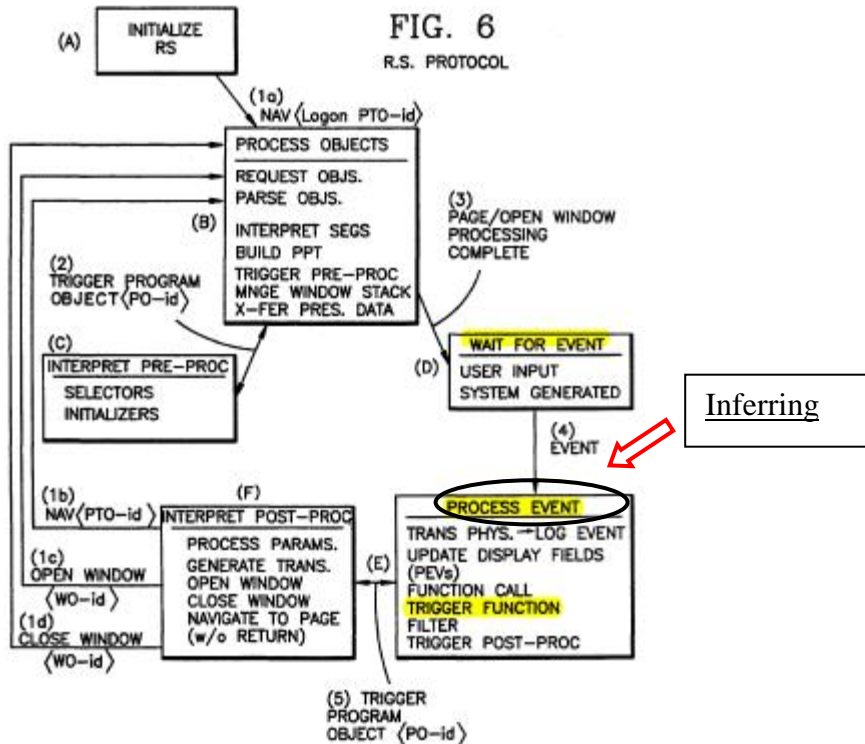
“The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the partitioned application.”

Col. 101 lines 53-54:

“receiving requests for partitioned applications at the reception system.”



inferring occurrence of the event and a context in which the event occurred based at least in part on the detected changes in state, and



Col. 14 lines 52-55:

“Program events will be specified in logical terms and will be mapped by the reception system to specific physical triggers...”

Col. 81 lines 43-51:

“Certain inputs, such as RETURN or mouse clicks in particular fields, are mapped to

logical events by keyboard manager 434, which are called completion (or commit) events. Completion events signify the completion of some selection or specification process associated with the partitioned application and trigger a partition level and/or page level post-processor to process the 'action' parameters associated with the user's selection and commit event."

Col. 39 lines 60-66:

"Reception system is aware of the occurrence of physical events during the...interactive sessions. When a physical event such as the depression of a ...key corresponds to a logical event such as the completion of data entry in a field..."

Col. 73 lines 52-64:

"Through this interaction, the user is able to input data into fields provided as part of the display, or may individually select choices causing a standard or personalized page to be built (as explained below) for display on the monitor of personal computer 405. Such inputs will cause RS 400 [reception system] to interpret events and trigger pre-processors or post-processors, retrieve specified objects, communicate with system components, control user options, cause the display of advertisements on a page, open or close window partitions to provide additional navigation possibilities, and collect and report data about events, including certain types of objects processed."

Col. 5, line 55 – Col. 6 line 9:

"Objects carry application programs and information for display at monitor screen 414 of RS 400. Application program objects, called pre-processor and post-processors, set up the environment for the user's interaction with network 10 and respond to events created when the user inputs information at keyboard 424 of RS 400. Such events typically trigger a program object to be processed, causing one of the following: sending of transactional information to the coapplications in one layer of the network 10; the receiving of information for use in programs or for presentation in application-dependent fields on monitor screen 414; or the requesting of a new objects to be processed by RS 400. Such objects may be part of the same application or a completely new application. The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the partitioned application.

Col. 6 lines 10-12:

RS 400 includes a means to communicate with network 10 to retrieve objects in response to events occurring at RS 400 and to send and receive messages.

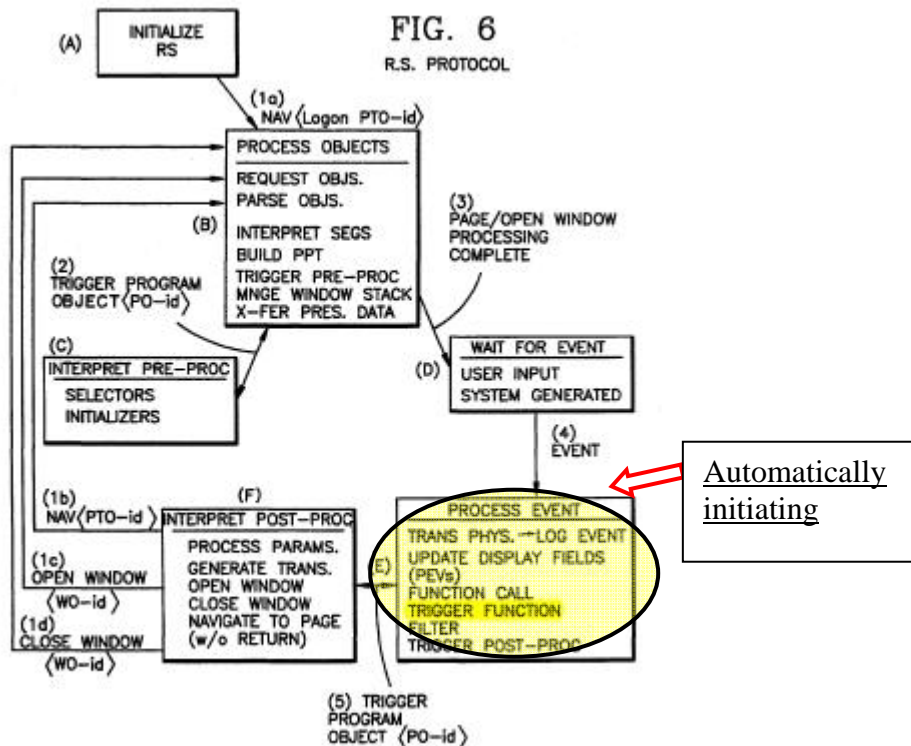
Col. 7 lines 35-46:

Objects may contain: control information; program instruction to set up an application processing environment and to process user or network created events; information about what is to be displayed and how it is to be displayed; references to programs to

	<p>be interpretively executed; and references to other objects, which may be called based upon certain conditions or the occurrence of certain events at the user's personal computer, resulting in the selection and retrieval of other partitioned applications packaged as objects.</p> <p>Col. 8 lines 2-8:</p> <p>If such objects are requested by the RS 400, the cache/concentrator 302 automatically requests the object from file server 205. The requested object is routed back to the requesting cache/concentrator 302, which automatically routes it to the communications line on which the request was originally made, from which it is received by the RS 400.</p> <p>Col. 9, lines 35-47:</p> <p>Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.</p> <p>Col. 81 lines 15-22:</p> <p>"This feature enables RS 400 to conditionally deliver information to the user base upon predetermined parameters, such as his personal demographics or locale. For example, the parameters specified may be the transaction codes required to retrieve the user's age, sex, and personal interest codes from records contained in user profiles stored at the switch/file server layer 200."</p>
link the inferred event with an action to be performed during the sales process based on prior sales experience using the sales system, and	<p>Col. 9 lines 30-38:</p> <p>"Advertisements 280 may be presented to the user on an individual basis from queues of advertisements that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400. Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications."</p> <p>Col. 73 lines 52-64:</p> <p>"Through this interaction, the user is able to input data into fields provided as part of the display, or may individually select choices causing a standard or personalized page to be built (as explained below) for display on the monitor of personal computer 405."</p>

Such inputs will cause RS 400 [reception system] to interpret events and trigger pre-processors or post-processors, retrieve specified objects, communicate with system components, control user options, cause the display of advertisements on a page, open or close window partitions to provide additional navigation possibilities, and collect and report data about events, including certain types of objects processed.”

automatically initiating an operation in one or more particular subsystems of the computer to facilitate a new action based on the inferred context.



Col. 74 lines 59-62:

“If the functions are part of [reception system] they can be altered or extended... [to] permit the execution of program objects to be triggered...”

Col. 8 lines 24-27:

“selecting another partitioned application to be processed upon a user generated completion event for the current partitioned application.”

Col. 99 lines 12-16:

The reception system comprising: “object processing means responsive to the input means for selectively retrieving and interpreting objects to extract data and program instructions for composing and generating the partitioned applications...”

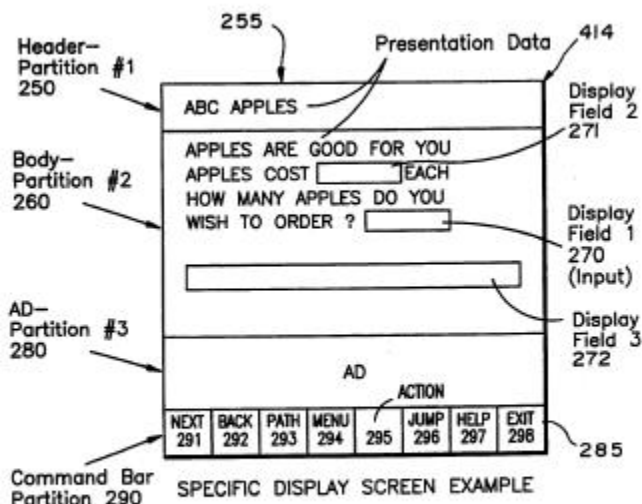


FIG. 3b

Col. 95 line 67- Col. 96 line 3:

“The page illustrated in FIG. 3(b) corresponds to a partitioned application that permit's [sic] a personal computer user to purchase apples. It shows how the monitor screen 414 of personal computer 405 might appear to the user.”

Col. 9, lines 35-47:

Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

US005347632A

United States Patent [19][11] **Patent Number:** **5,347,632****Filepp et al.**[45] **Date of Patent:** **Sep. 13, 1994****[54] RECEPTION SYSTEM FOR AN
INTERACTIVE COMPUTER NETWORK
AND METHOD OF OPERATION**

[75] Inventors: Robert Filepp, Springfield, N.J.;
Michael L. Gordon, Dobbs Ferry,
N.Y.; Alexander W. Bidwell, New
York, N.Y.; Francis C. Young, Pearl
River, N.Y.; Allan M. Wolf,
Ridgefield, Conn.; Sam Meo, New
York, N.Y.; Duane Tiemann,
Trumbull; Robert D. Cohen, New
Fairfield, both of Conn.; Mel Bellar,
New York, N.Y.; Kenneth H.
Appleman, Brooklyn, N.Y.; Lawrence
Abrahams, New York, N.Y.; Michael
J. Silfen, Croton, N.Y.

[73] Assignee: Prodigy Services Company, White
Plains, N.Y.

[21] Appl. No.: 388,156

[22] Filed: Jul. 28, 1989

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 328,790, Mar. 23,
1989, abandoned, which is a continuation-in-part of
Ser. No. 219,931, Jul. 15, 1988, abandoned.

[51] Int. Cl.⁵ G06F 13/38

[52] U.S. Cl. 395/200; 364/DIG. 1;
364/284.4; 364/228.4; 364/246.3

[58] Field of Search 364/DIG. 1, DIG. 2,
364/DIG. 1 MS File; 395/200, 325; 235/379,
382

[56] References Cited**U.S. PATENT DOCUMENTS**

3,653,001 3/1972 Ninke 364/DIG. 1
4,091,448 5/1978 Clausen 364/DIG. 1
4,200,930 4/1980 Rawlings et al. 364/DIG. 1

4,289,930 9/1981 Connolly et al. 179/2
4,319,336 3/1982 Anderson et al. 364/DIG. 1
4,460,960 7/1984 Anderson et al. 364/DIG. 1
4,553,222 11/1985 Kurland et al. 364/900
4,575,679 3/1986 Simon et al. 178/4
4,691,340 9/1987 Maeda et al. 379/96
4,724,521 2/1988 Carron et al. 364/300
4,751,669 6/1988 Sturgis et al. 364/900
4,787,050 11/1988 Suzuki 364/479
4,805,119 2/1989 Maeda et al. 364/518
4,805,134 2/1989 Calo et al. 364/900
4,851,994 7/1989 Toda et al. 364/200
4,882,705 11/1989 Yasue et al. 364/900

OTHER PUBLICATIONS

Gecsei, Jan, *The Architecture of Videotex Systems*, Pren-
tice-Hall, Inc., Mar. 25, 1983.

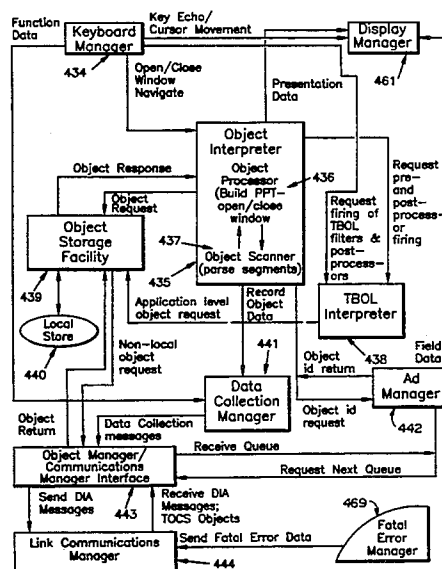
Date, "An Introduction to Database Systems", Ad-
dison-Wesley, 1983, pp. 274-279, 291-340.

Primary Examiner—Thomas M. Heckler

Attorney, Agent, or Firm—Paul C. Scifo

[57] ABSTRACT

An interactive computer system network enables a user to display desired information, such as news, financial and cultural information, and perform desired transactional services, such as banking and shopping, through any of a plurality of types of personal computers. User inputs are received by the personal computer and are translated into personal computer-independent data objects and executable code objects which are then processed by the network. These objects comprise partitioned applications required to process user inputs, portions of which are distributed and stored either locally within the personal computer or remotely in a host computer. User characteristics are monitored by the system in order to generate and display specific advertisements to the user based on individual usage characteristics and predetermined interests.

43 Claims, 16 Drawing Sheets

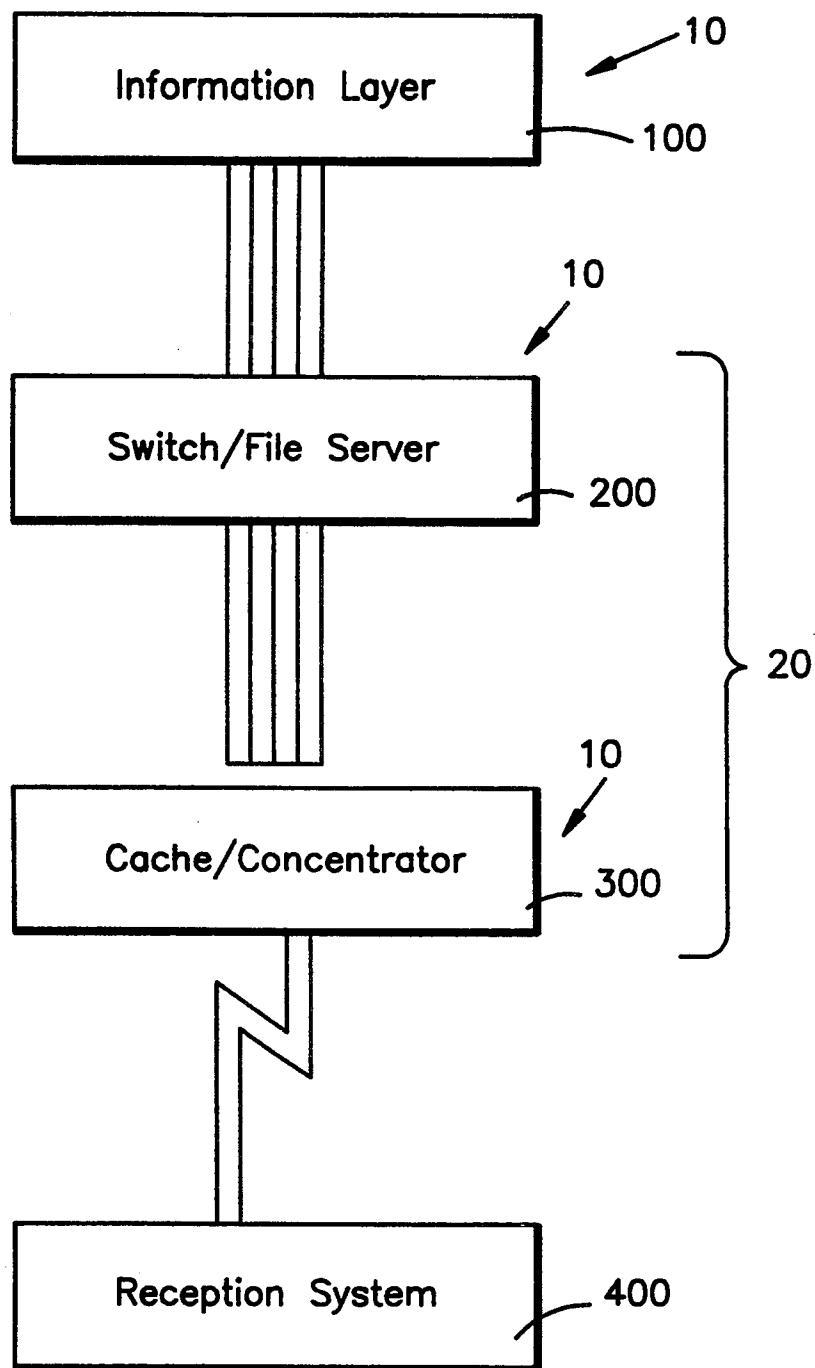
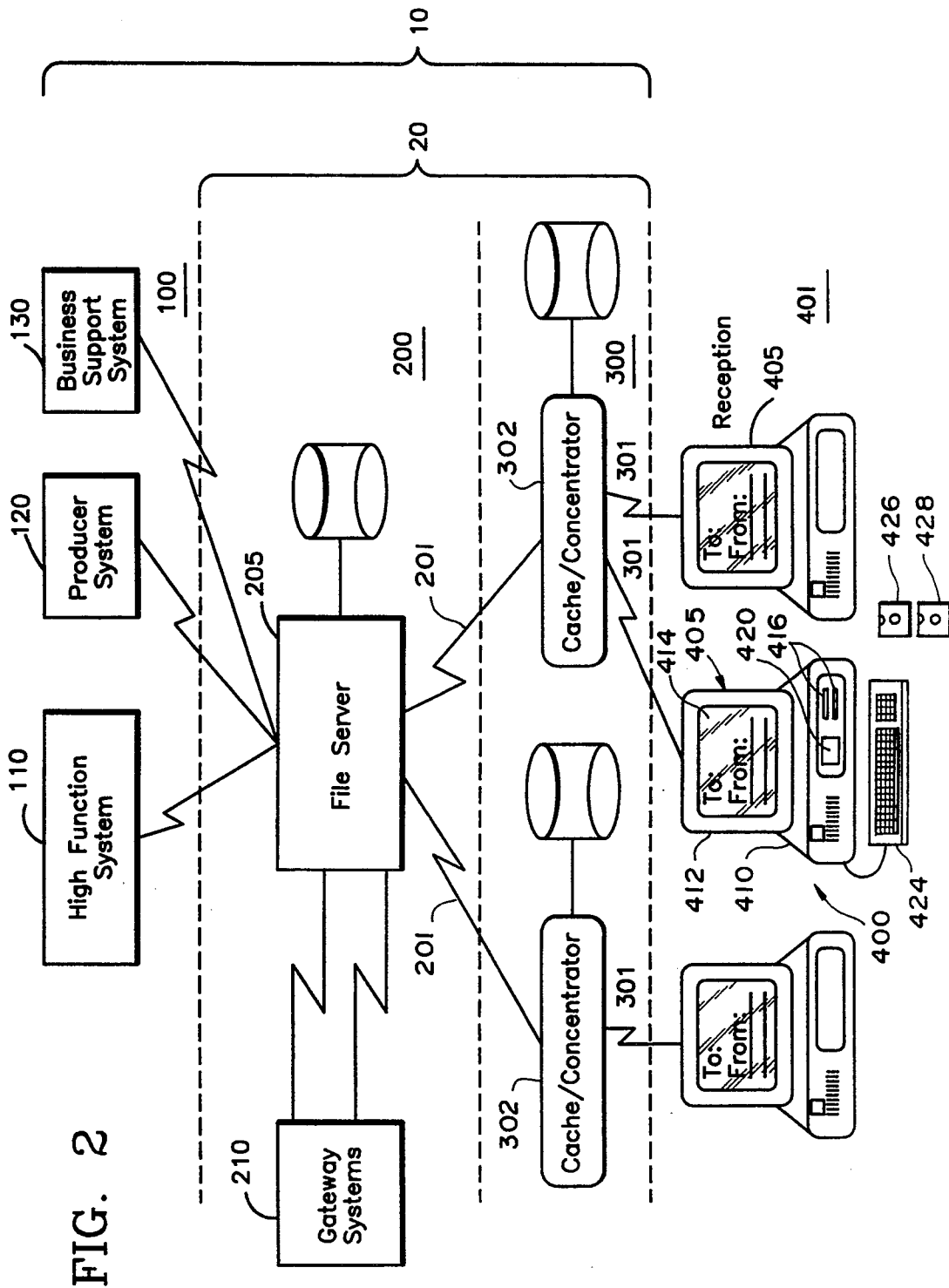


FIG. 1



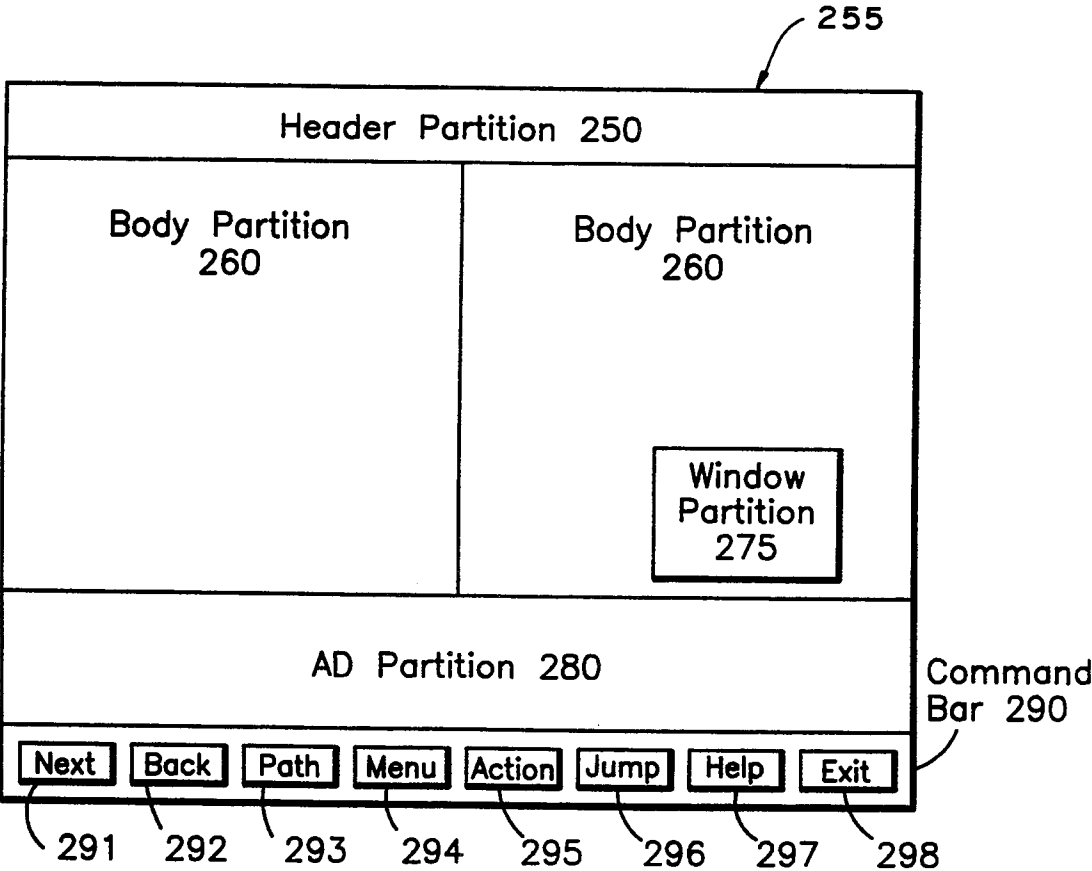


FIG. 3a

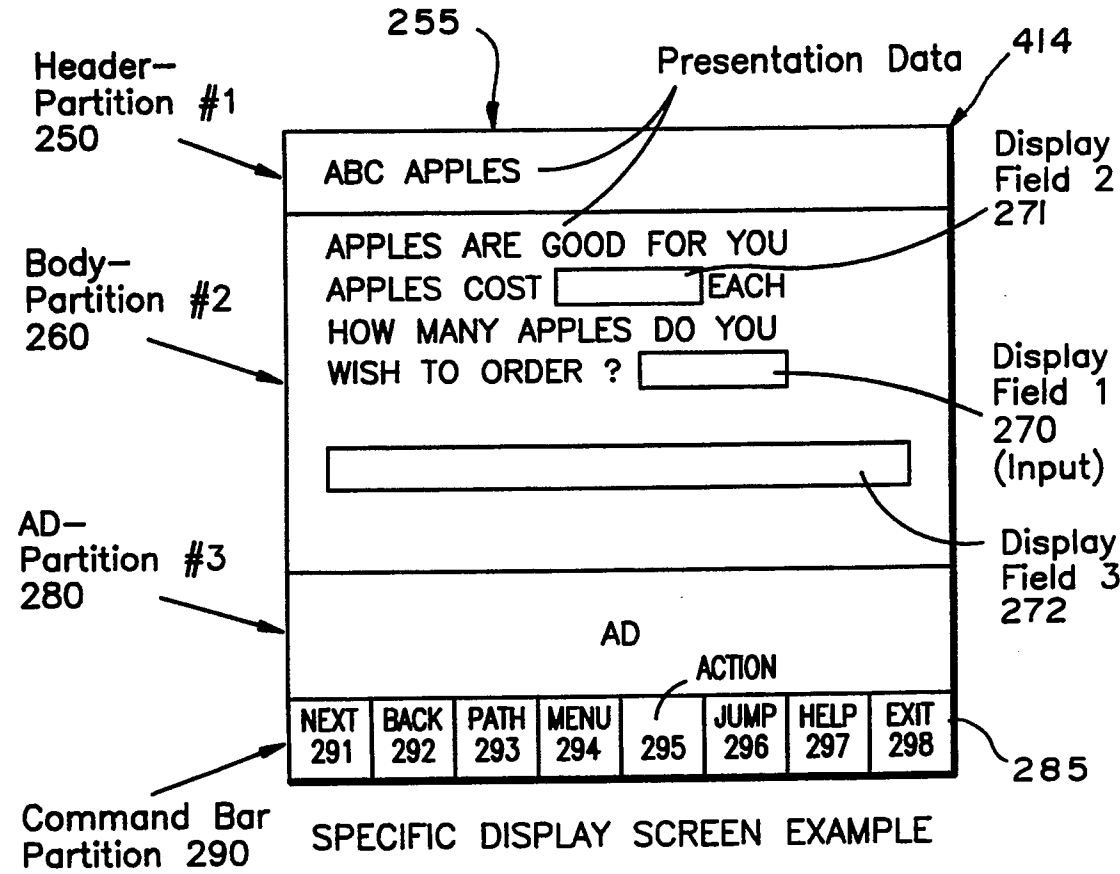


FIG. 3b

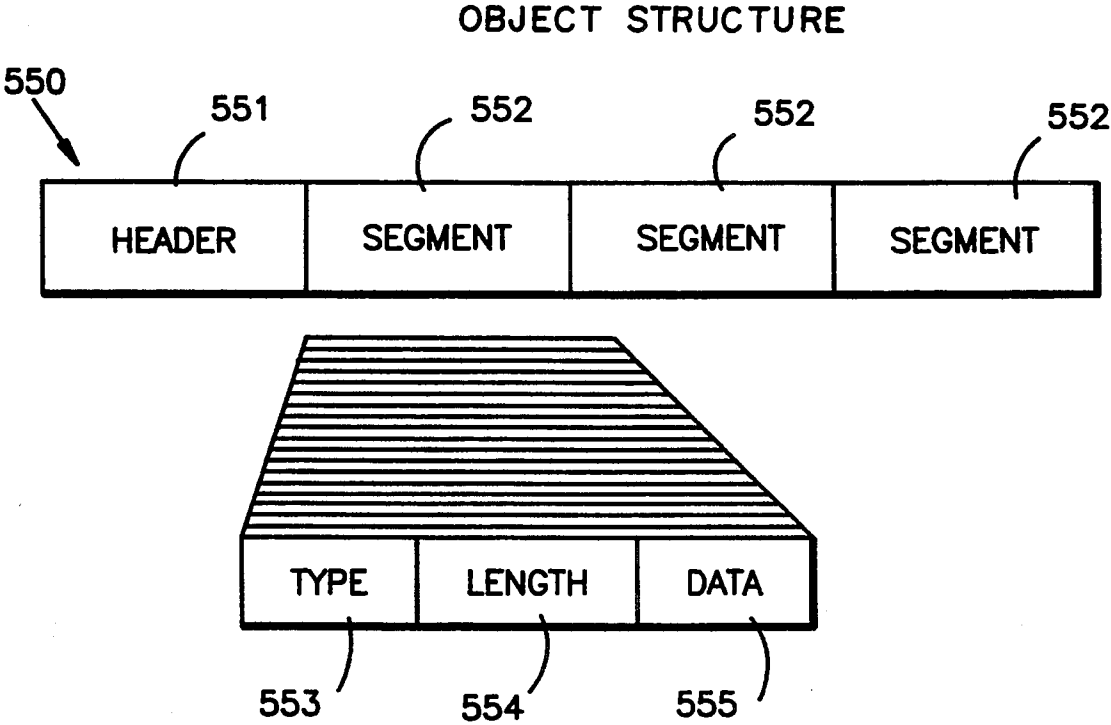


FIG. 4a

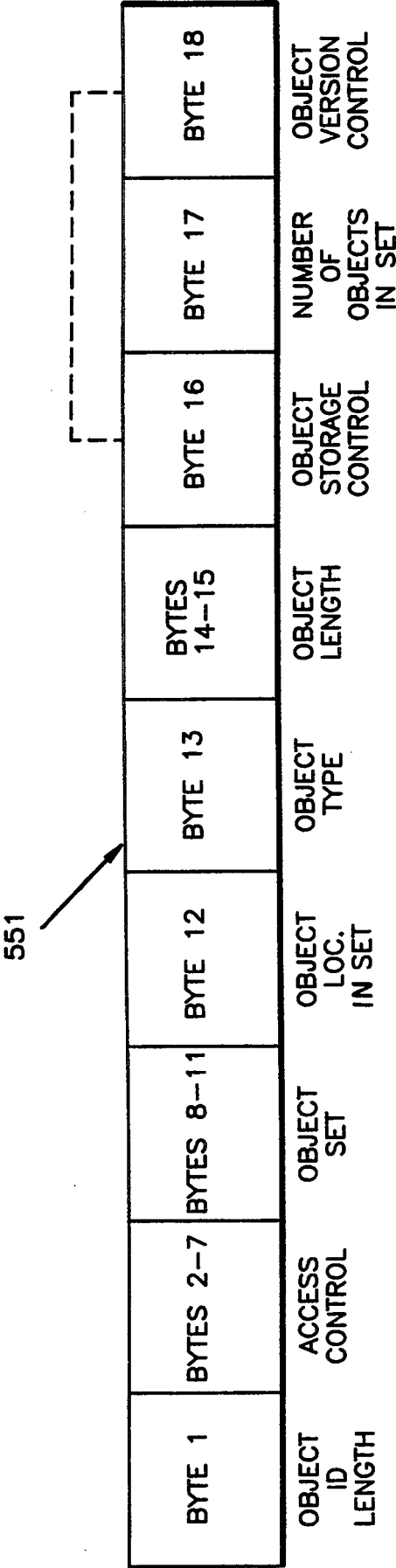


FIG. 4b

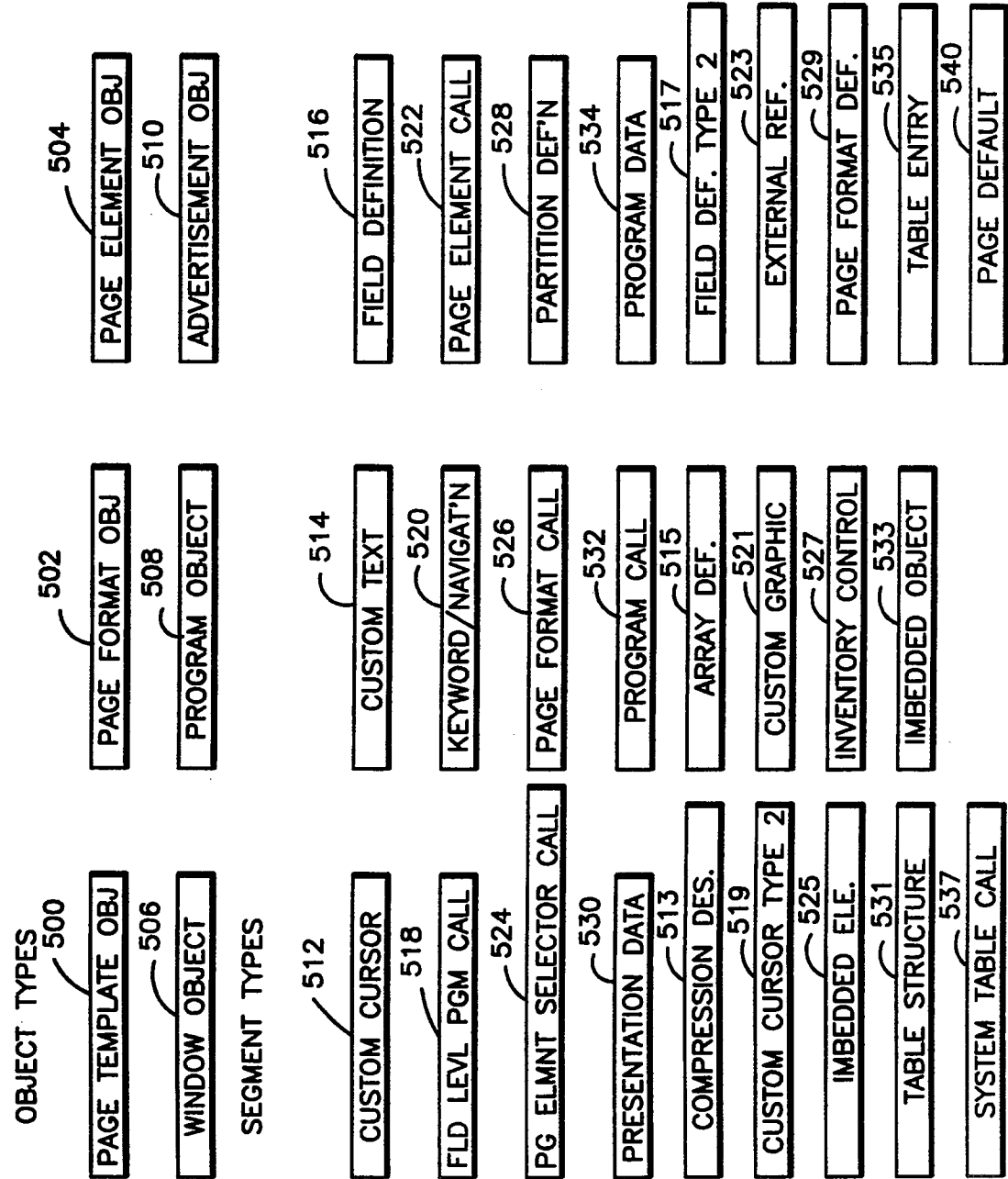


FIG. 4c

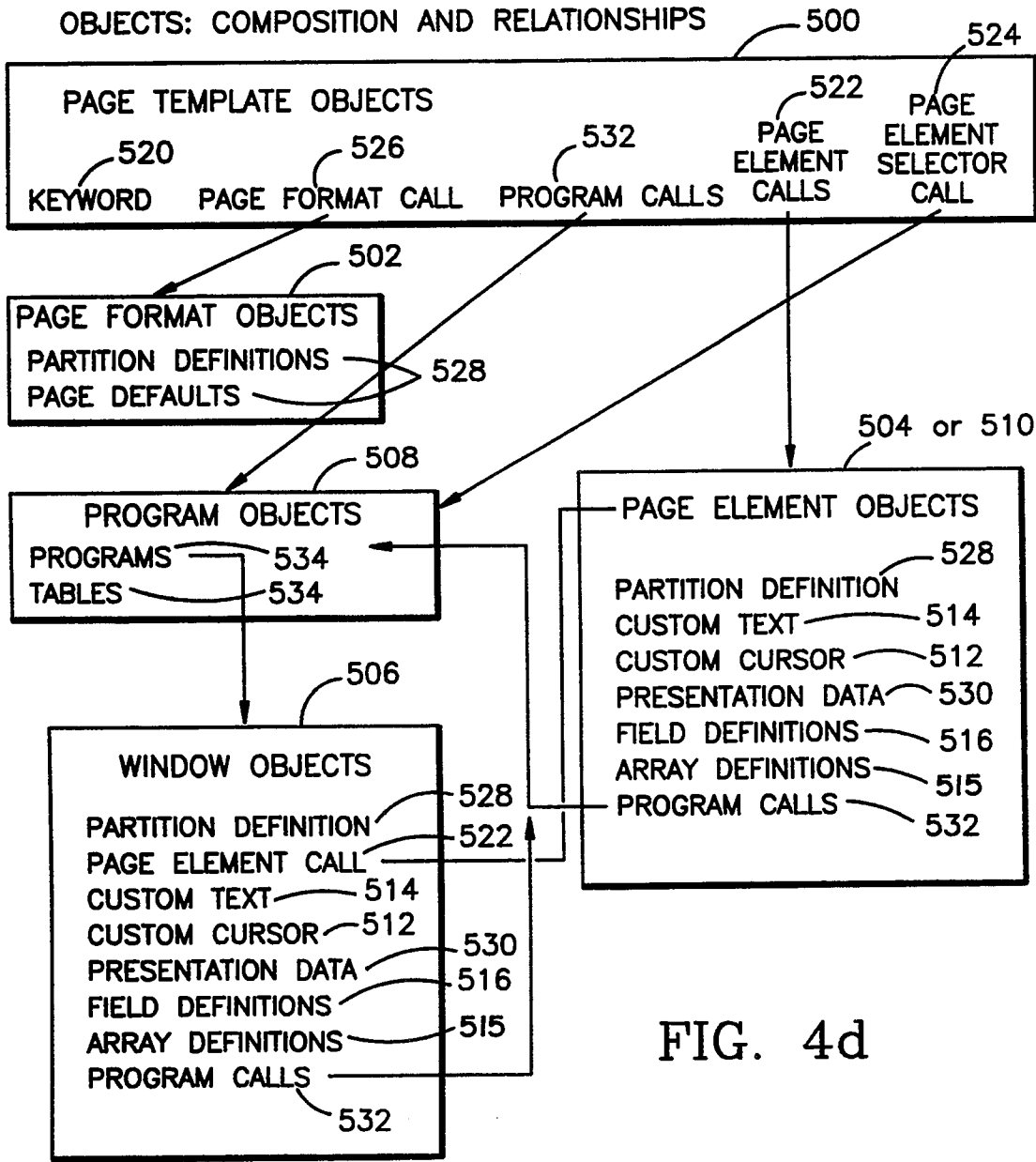


FIG. 4d

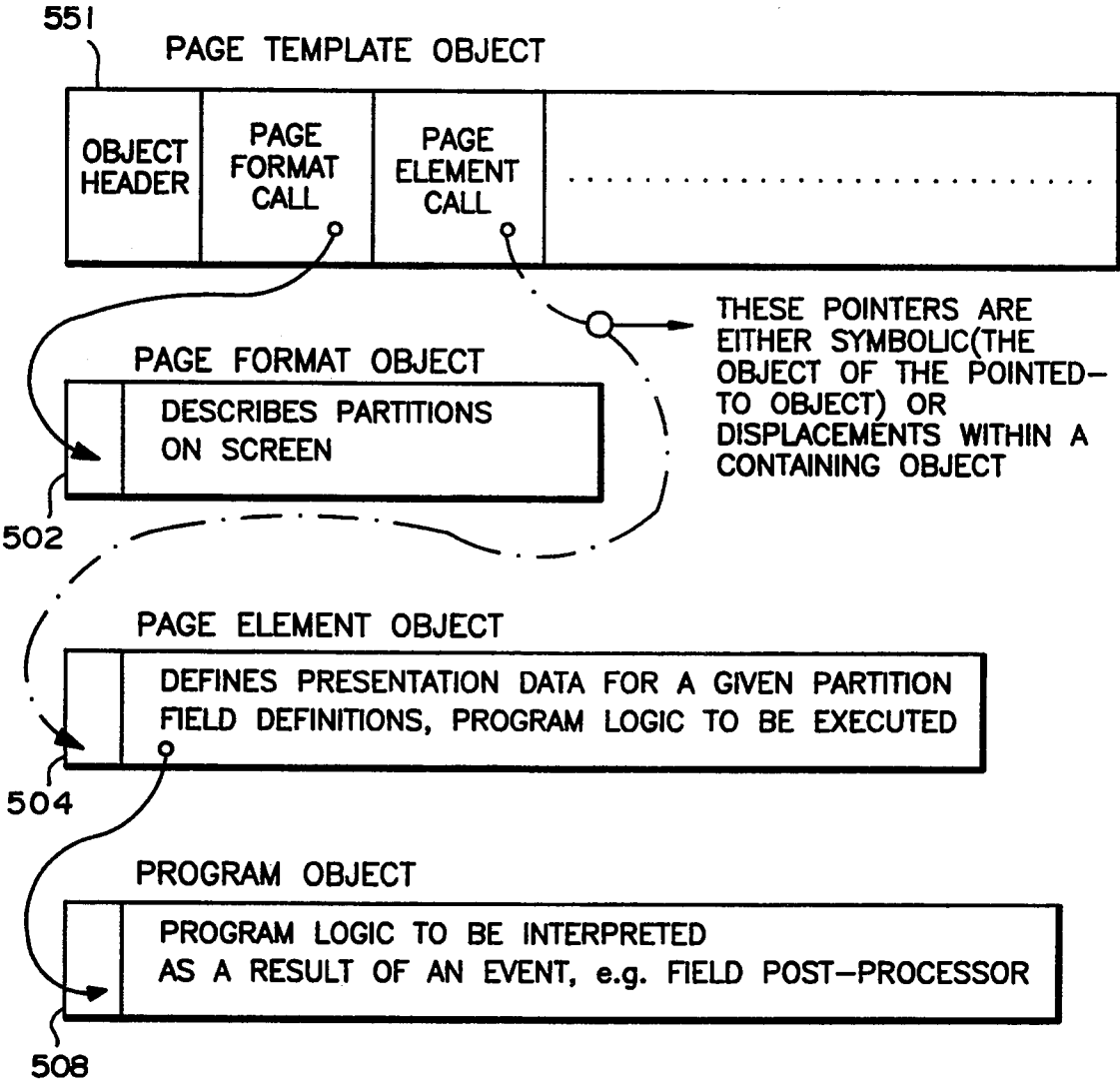
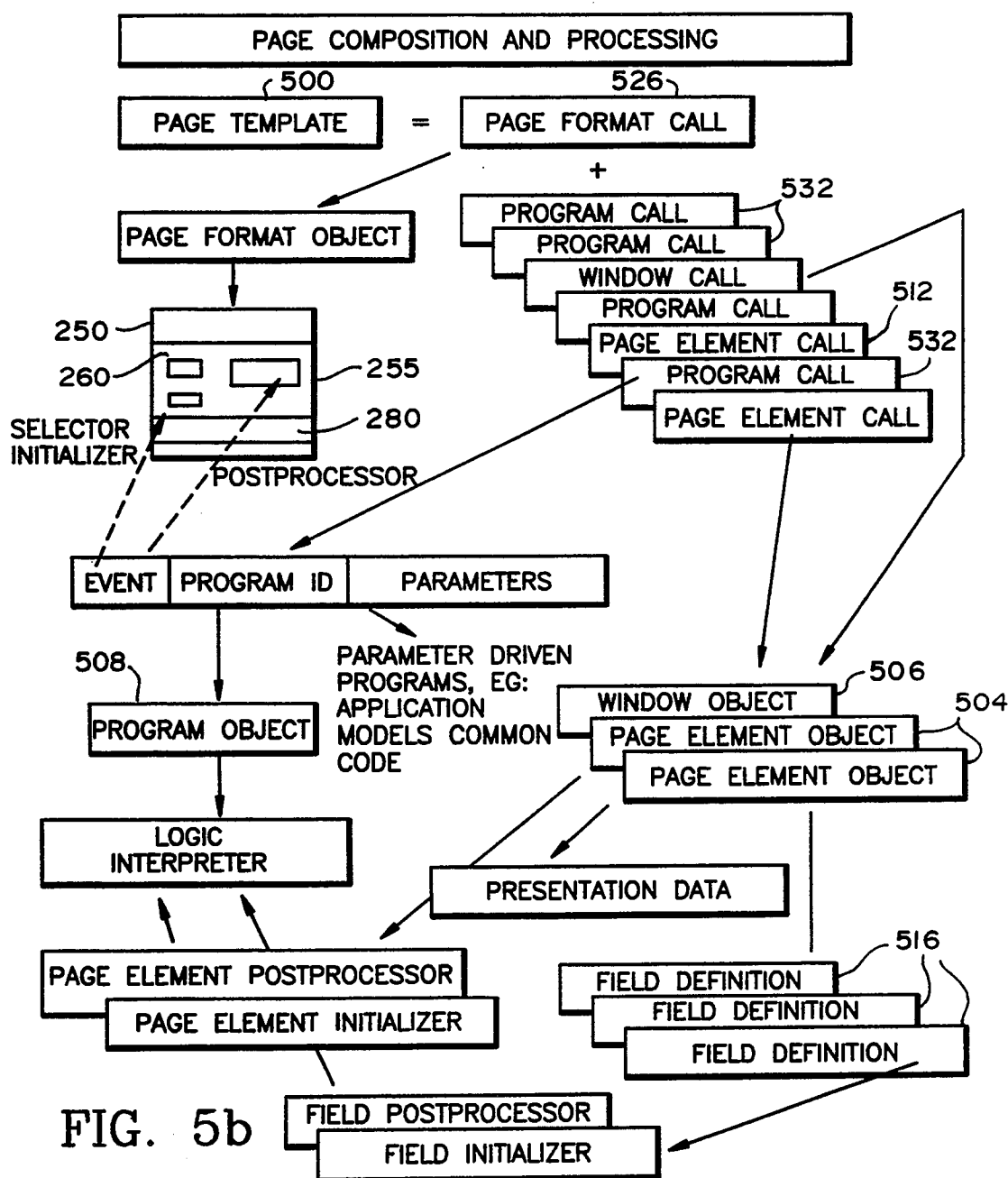
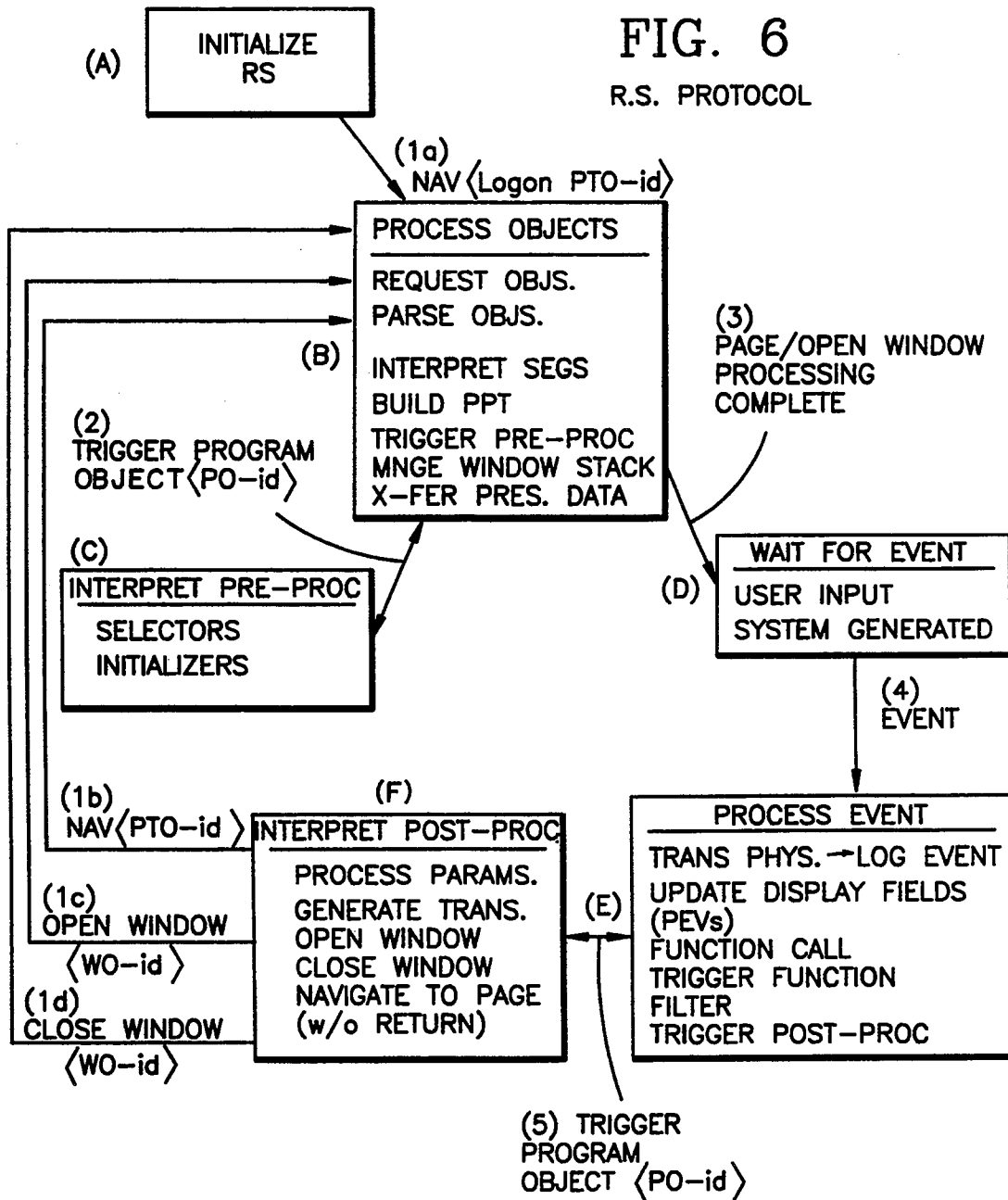
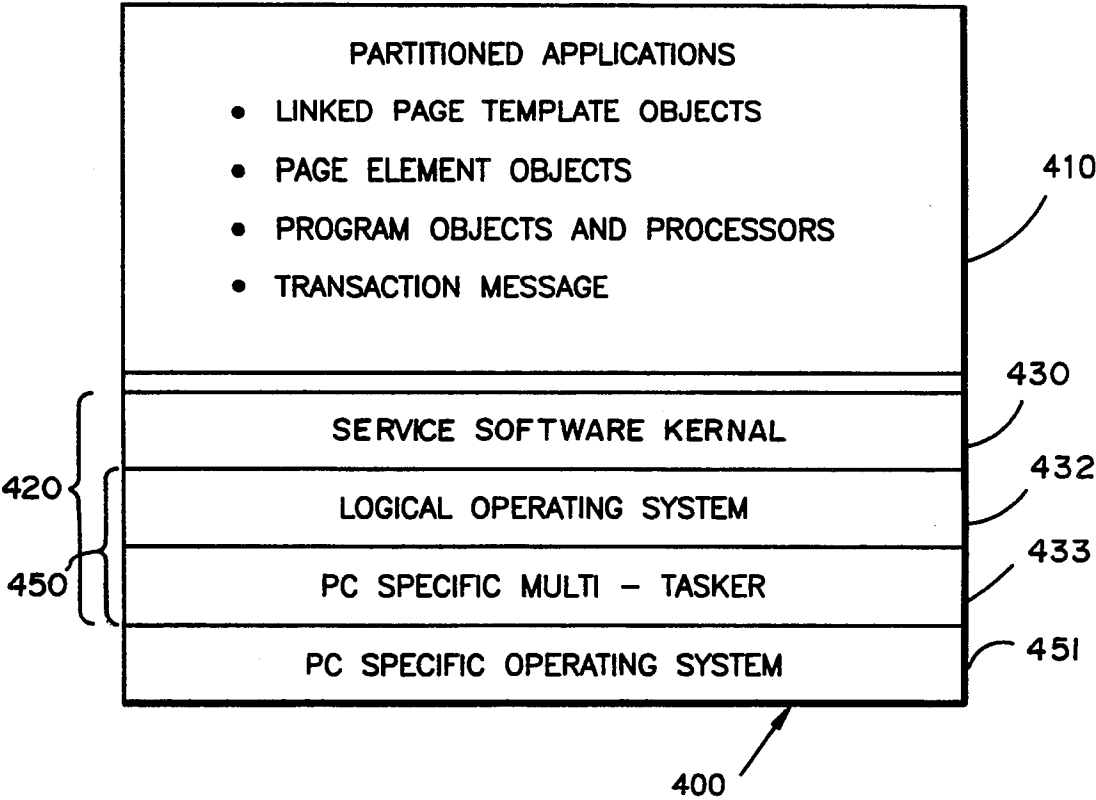


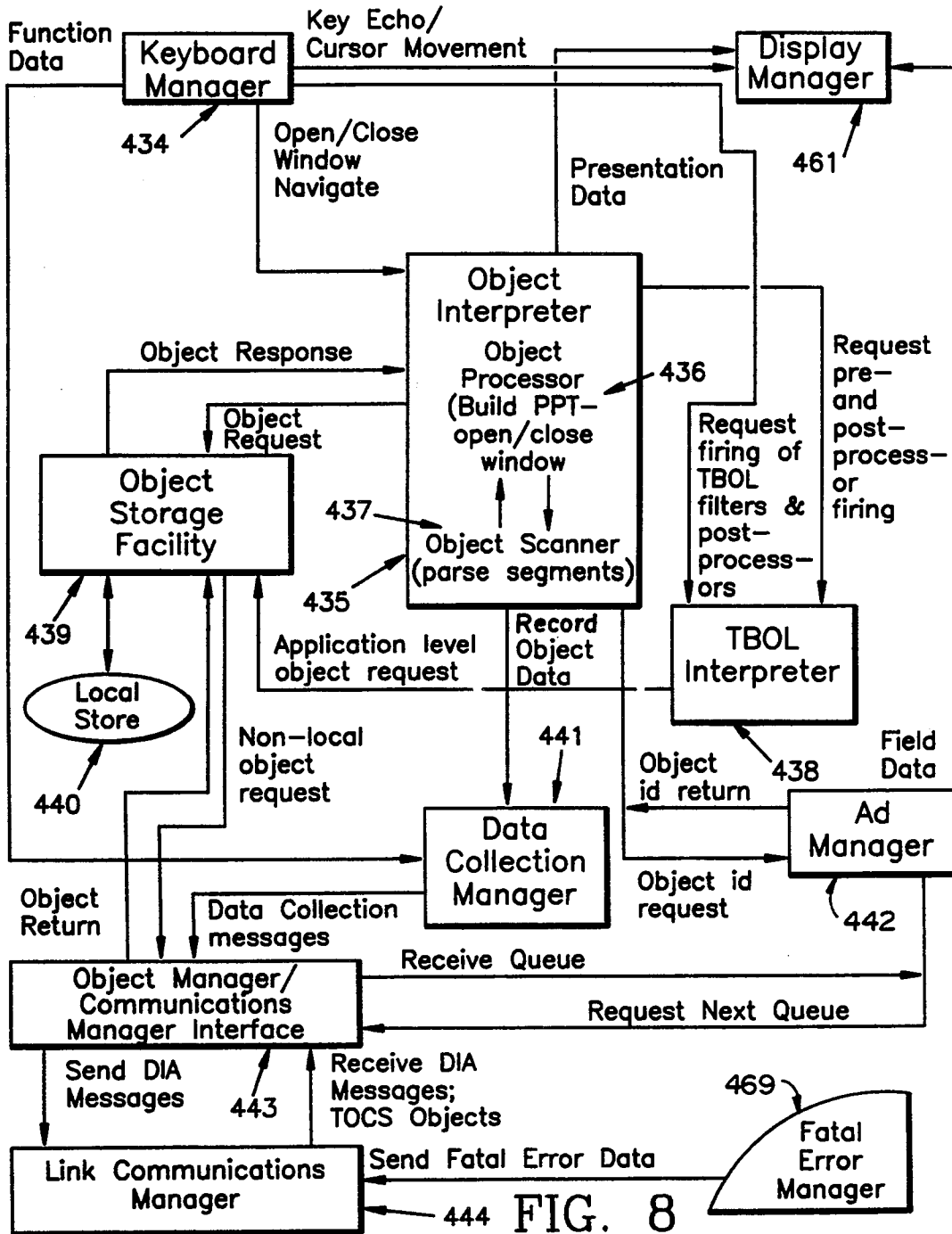
FIG. 5a







RECEPTION SYSTEM LAYERS
FIG. 7



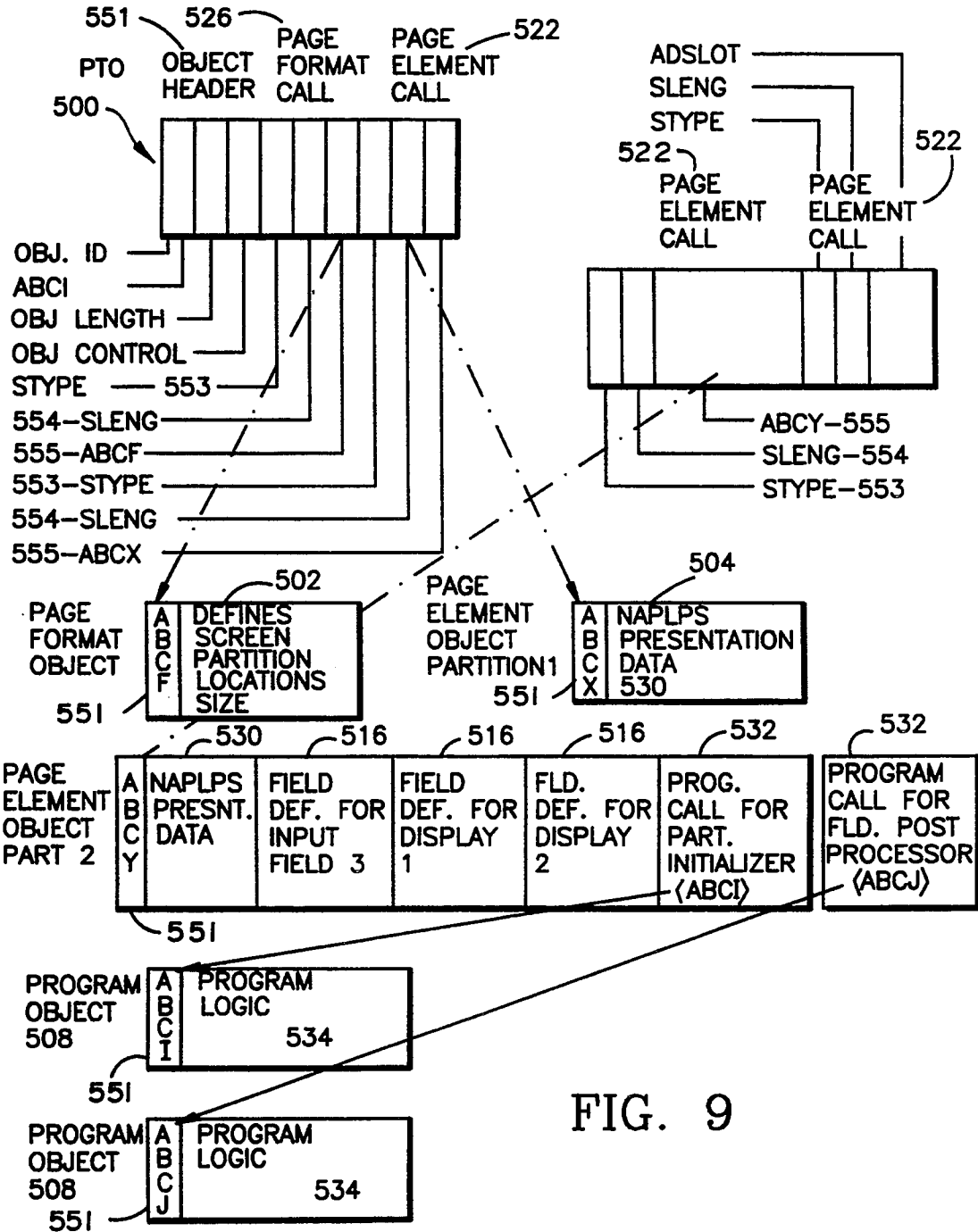


FIG. 9

FIG. 10
PAGE PROCESSING TABLE (ppt)

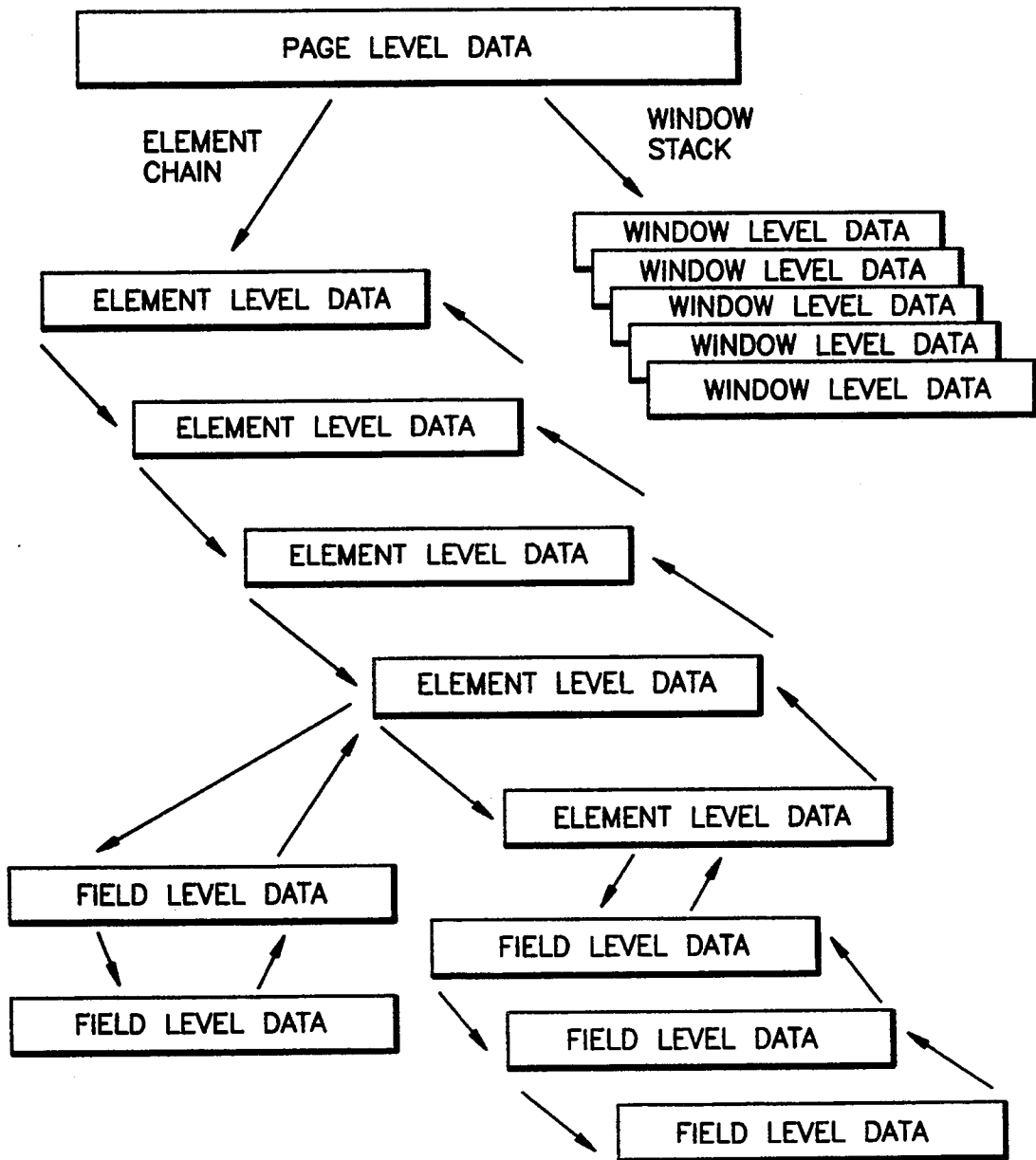
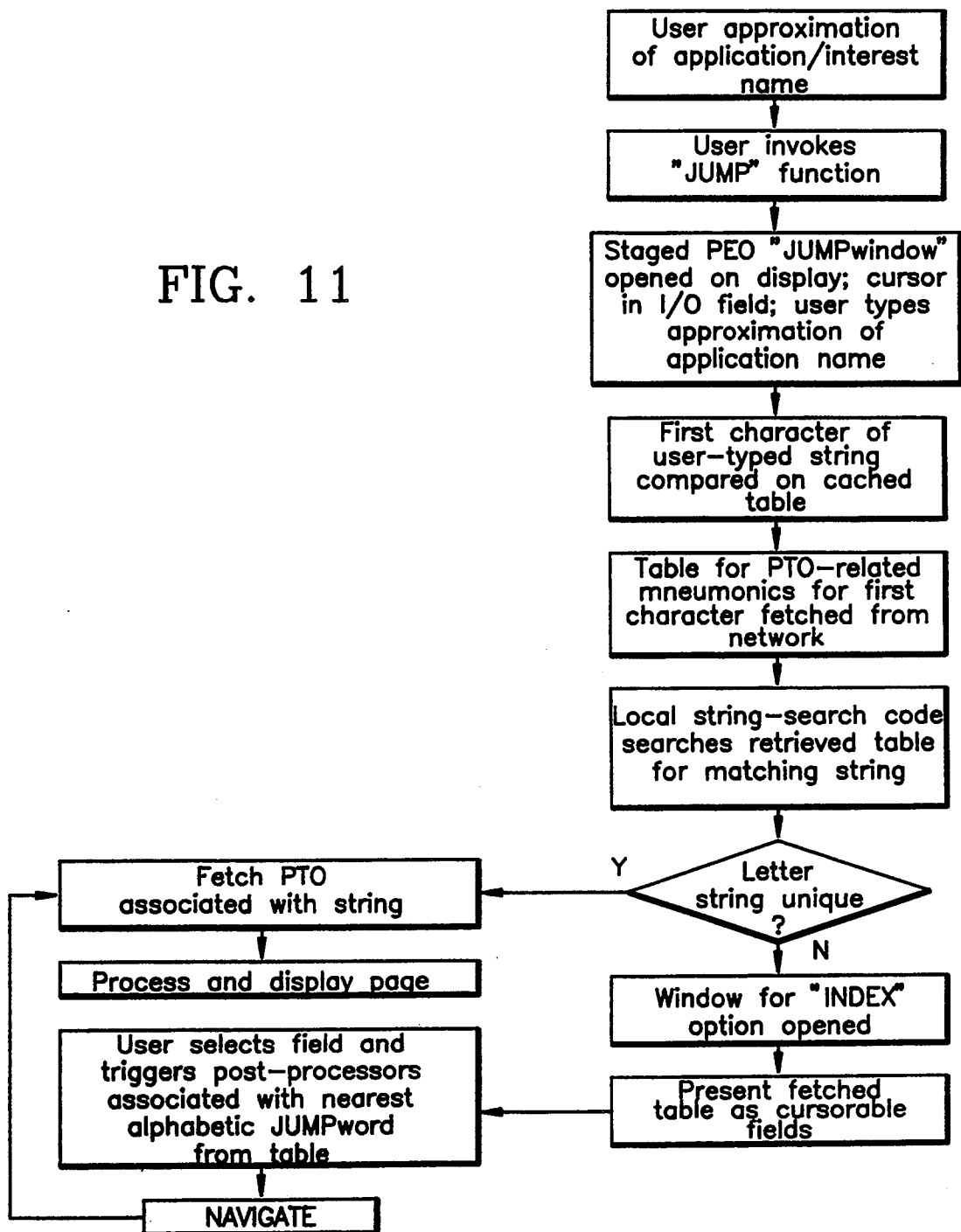


FIG. 11



5,347,632

1

RECEPTION SYSTEM FOR AN INTERACTIVE COMPUTER NETWORK AND METHOD OF OPERATION

BACKGROUND OF THE INVENTION

This is a continuation-in-part of the application Ser. No. 328,790, filed Mar. 23, 1989, which itself was a continuation in part of the application Ser. No. 219,931, filed Jul. 15, 1988, both now abandoned.

This invention relates generally to a distributed processing, interactive computer network intended to provide very large numbers of simultaneous users; e.g. millions, with access to a large number; e.g., thousands, of applications which include pre-created, interactive text/graphic sessions; and more particularly, to a computer network in which the interactive text/graphic sessions are comprised of pre-created blocks of data and program instructions which may be distributed downwardly in the network for use at a software enhanced user computer terminal that reduces processing demand on the higher-level network elements, thus permitting the higher-level elements to function primarily as data supply and maintenance resource, and, thereby, reduce network complexity, cost and response time.

Interactive computer networks are not new. Traditionally they have included conventional, hierarchical architectures wherein a central, host computer responds to the information requests of multiple users. An illustrative example would be a time-sharing network in which multiple users, each at a remote terminal, log onto a host computer having data and software resource that sequentially receives the user's data processing requests, executes them and supplies responses back to the users.

While such networks have made the processing power of large computers available to many users, problems have existed with them. Particularly, in such networks, the host has been required to satisfy all the user data processing requests. As a result, processing bottle-necks arise at the host that tax the host resources causing slowdowns in network response time and requiring expansion in computing power; i.e., bigger and more complex computer facilities, where acceptable response times are sought to be maintained in the face of increases in the number of users to be served.

The size and complexity of the network host, however, is particularly critical in the case of commercial interactive computer network recently introduced to offer large number of the general public text and graphics information that enable not only at home shopping and financial management such as banking and bill paying, but also the providing of information relating to entertainment, business and personal matters.

As can be appreciated, in such state of the art information and shopping networks, the network must be able to provide the information and shopping services with a minimal amount of network resources in order to maintain the capital investment in the network at a level that renders the services economical to use. Unlike military and governmental networks where, because of the nature of the service provided, capital investment is a secondary concern, in commercial information and shopping services, the capital investment in the network resources must be kept low in order to make the network affordable both to the users and those who would rely on the network as a channel of distribution for their goods and/or services. Further, in addition, to main-

2

taining capital investment at a minimum, it is also desirable to maintain network response time at a minimum in order to not only capture and hold the user's attention, but also quickly free the network to satisfy the requests of other users. As noted, this ability to satisfy requests with minimal network resources is required to enable the network to serve large numbers of users and, thereby, render the network economical.

While conventional, previously known time-sharing network designs have attempted to alleviate host complexity and response time problems by providing some processing at the user site; i.e., "smart terminals", still the storage of the principal data and software resources needed for processing at the host continues to create a burden on network complexity and response time which renders the conventional approach unsuited for the large numbers of users required for a commercially viable computer based information and shopping network.

SUMMARY OF INVENTION

Accordingly, it is an object of this invention to provide method and apparatus which permit a very large number of users to obtain access to a large number of applications which include interactive text/graphic sessions that have been created to enable the users to obtain information and transactional services.

It is a further object of this invention to provide method and apparatus which permit the data and programs necessary to support applications including interactive text/graphic sessions to be distributed over a computer network.

It is a still further object of this invention to provide software that will enable a conventional personal computer to be coupled to a computer network to establish a reception system suitable for supporting applications which include interactive text/graphic sessions created to enable the user to obtain information and conduct shopping events.

It is yet another object of this invention to provide method and apparatus that would permit information and transactional services to be provided to users based upon predetermined parameters such as user demographics and/or locale.

It is yet another object of the invention to provide method and apparatus capable of collecting data regarding usage of the network and to condition the applications and the included text/graphics sessions based upon the reactions to the applications by the users.

Briefly, to achieve the above and other objects and features, the invention includes method and apparatus for providing interactive applications containing text and graphics at the monitor of a personal computer, that has been configured as a reception system by the inclusion and running of reception system software that enables the reception system so formed to be electronically connected to a network specially adapted to create, maintain and supply databases and portions thereof containing the applications. In accordance with its method aspects, the invention includes procedures for formulating objects that have been specially structured to include display data, control data and program instructions for supporting the applications at the network reception systems, the objects being pre-created, parceled units of information that may be distributed and stored at lower levels in the network; e.g., at the reception system, so as to reduce processing demand on

5,347,632

3

the network higher element, and thereby permit the higher elements to function primarily as elements for maintaining and supplying the database information.

Further, in preferred form, the method aspect of the invention, features use, of specially structured messages that harmonize and facilitate communications between the different elements of the network and computing elements external to the network that may be called upon to supply information to support the applications.

Also in preferred form, the method aspect of the invention features specially prepared program instructions within the objects that permit the objects to be executed at the reception system in conjunction with the application software.

Also in preferred form, the invention includes procedures in the form of application software that contain modules that individually and in combination facilitate the execution of objects and the handling of messages at the reception system so that the interactive sessions may be supported at the reception system.

Still further in its apparatus, aspects the invention includes a reception system comprised of one of a plurality of brands of personal computers combined with the application software for use in the interactive network for displaying information and providing transactional services to a user. In preferred form, the reception system further comprises input means for receiving user inputs; storage means for storing objects containing data or interpretively executable programs, the objects comprising a plurality of partitioned applications; and object processing means, responsive to the input means, for selectively retrieving objects from the storage means and interpreting and executing the partitioned applications.

DESCRIPTION OF THE DRAWINGS

The above and further objects, features and advantages of the invention will become clear from the following more detailed description when read with reference to the accompanying drawings in which:

FIG. 1 is a block diagram of the interactive computer network in accordance with the invention;

FIG. 2 is a schematic diagram of the network illustrated in FIG. 1;

FIGS. 3a and 3b are plan views of a display screen presented to a user in accordance with the invention;

FIGS. 4a, 4b, 4c and 4d are schematic drawings that illustrate the structure of objects, and object segments utilized within the interactive network in accordance with the invention;

FIG. 5a is a schematic diagram that illustrates the configuration of the page template object in accordance with the invention;

FIG. 5b is a schematic diagram that illustrates page composition in accordance with the invention;

FIG. 6 is a schematic diagram that illustrates the protocol used by the reception system to support user applications in accordance with the invention;

FIG. 7 is a schematic diagram that illustrates major layers of the reception system in accordance with the invention;

FIG. 8 is a block diagram that illustrates native code modules of the reception system in accordance with the invention;

FIG. 9 is a schematic diagram that, illustrates an example of a partitioned application to be processed by the reception system in accordance with the invention;

4

FIG. 10 illustrates generation of a page with a page processing table in accordance with the invention; and

FIG. 11 is a flow diagram for an aspect of the navigation method in accordance with the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT GENERAL SYSTEM DESCRIPTION

With reference to FIGS. 1, 2, the invention includes a plurality of reception units within reception layer 401 of interactive computer network 10 for displaying information and providing transactional services. In this arrangement, many users each accesses network 10 with a conventional personal computer; i.e., one of the IBM or IBM-compatible type, which has been provided with applications software in accordance with a preferred form of the invention to constitute a reception system (RS) 400.

As shown in FIG. 1, interactive network 10 uses a layered structure that includes an information layer 100, a switch/file server layer 200, and cache/concentrator layer 300 as well as reception layer 401. This structure maintains active application databases and delivers requested parts of the databases on demand to the plurality of RS 400's, shown in FIG. 2. As seen in FIG. 2, cache/concentrator layer 300 includes a plurality of cache/concentrator units 302, each of which serve a plurality of RS 400 units over lines 301. Additionally, switch/file server layer 200 is seen to include a server unit 205 connected to multiple cache/concentrator units 302 over lines 201. Still further, server unit 205 is seen to be connected to information layer 100 and its various elements, which act as means for producing, supplying and maintaining the network databases and other information necessary to support network 10. Continuing, switch/file layer 200 is also seen to include gateway systems 210 connected to server 205. Gateways 210 couple layer 200 to other sources of information and data; e.g., other computer systems. As will be appreciated by those skilled in the art, layer 200, like layers 401 and 300 could also include multiple servers, gateways and information layers in the event even larger numbers of users were sought to be served.

Continuing with reference to FIG. 2, each RS 400 is seen to include a personal computer 405 having a CPU 410 including a microprocessor (as for example the type made by INTEL Corporation in its X'86 family of microprocessors), companion RAM and ROM memory and other associated elements, monitor 412 with screen 414 and a keyboard 424. Further, personal computer 405 may also include one or two floppy disk drives 416 for receiving diskettes 426 containing application software in accordance with this invention for supporting the interactive sessions with network 10 and diskettes 428 containing operating systems software; i.e., MS-DOS, suitable for the personal computer 405 being used. Personal computer 405 may also include a hard-disk drive 420 for storing the application software and operating system software which may be transferred from diskettes 426 and 428 respectively.

Once so configured, each RS 400 provides: a common interface to other elements of interactive computer network 10; a common environment for application processing; and a common protocol for user application conversation which is independent of the personal computer brand used. RS 400 thus constitutes a universal terminal for which only one version of all applications on network 10 need be prepared, thereby rendering the

5,347,632

5

applications interpretable by a variety of brands of personal computers of the IBM or IBM-compatible type.

RS 400 formulated in this fashion is capable of communication with the host system to receive information containing either of two types of data, namely objects and messages. Objects have a uniform, self-defining format known to RS 400, and include data types, such as interpretable programs and presentation data for display at monitor screen 414 of the user's personal computer. Applications presented at RS 400 are partitioned into objects which represent the minimal units available from the higher levels of interactive network 10 or RS 400. In this arrangement, each application partition typically represents one screen or a partial screen of information, including fields filled with data used in transactions with network 10. Each such screen, commonly called a page, is represented by its parts and is described in a page template object, discussed below.

Applications, having been partitioned into minimal units, are available from higher elements of network 10 or RS 400, and are retrieved on demand by RS 400 for interpretive execution. Thus, not all partitions of a partitioned application need be resident at RS 400 to process a selected partition, thereby raising the storage efficiency of the user's RS 400 and minimizing response time. Each application partition is an independent, self-contained unit and can operate correctly by itself. Each partition may refer to other partitions either statically or dynamically. Static references are built into the partitioned application, while dynamic references are created from the execution of program logic using a set of parameters, such as user demographics or locale. Partitions may be chosen as part of the RS processing in response to user created events, or by selecting a key word of the partitioned application (e.g., "JUMP" or "INDEX," discussed below), which provides random access to all services represented by partitioned applications having key words.

Objects provide a means of packaging and distributing partitioned applications. As noted, objects make up one or more partitioned applications, and are retrieved on demand by a user's RS 400 for interpretive execution and selective storage. All objects are interpreted by RS 400, thereby enabling applications to be developed independently of the personal computer brand used.

Objects may be nested with one another or referenced by an object identifier (object-id) from within their data structure. References to objects permit the size of objects to be minimized. Further, the time required to display a page is minimized when referenced objects are stored locally at RS 400 (which storage is determined by prior usage meeting certain retention criteria), or have been pre-fetched, or in fact, are already used for the current page.

Objects carry application programs and information for display at monitor screen 414 of RS 400. Application program objects, called pre-processor and post-processors, set up the environment for the user's interaction with network 10 and respond to events created when the user inputs information at keyboard 424 of RS 400. Such events typically trigger a program object to be processed, causing one of the following: sending of transactional information to the coapplications in one layer of the network 10; the receiving of information for use in programs or for presentation in application-dependent fields on monitor screen 414; or the requesting of a new objects to be processed by RS 400. Such

6

objects may be part of the same application or a completely new application.

The RS 400 supports a protocol by which the user and the partitioned applications communicate. All partitioned applications are designed knowing that this protocol will be supported in RS 400. Hence, replication of the protocol in each partitioned application is avoided, thereby minimizing the size of the partitioned application.

RS 400 includes a means to communicate with network 10 to retrieve objects in response to events occurring at RS 400 and to send and receive messages.

RS 400 includes a means to selectively store objects according to a predetermined storage criterion, thus enabling frequently used objects to be stored locally at the RS, and causing infrequently used objects to forfeit their local storage location. The currency of objects stored locally at the RS 400 is verified before use according to the object's storage control parameters and the storage criterion in use for version checking.

Selective storage tailors the contents of the RS 400 memory to contain objects representing all or significant parts of partitioned applications favored by the user. Because selective storage of objects is local, response time is reduced for those partitioned applications that the user accesses most frequently.

Since much of the application processing formerly done by a host computer in previously known time-sharing networks is now performed at the user's RS 400, the higher elements of network 10, particularly layer 200 has as its primary functions the routing of messages, serving of objects, and line concentration. The narrowed functional load of the higher network elements permits many more users to be serviced within the same bounds of computer power and I/O capability of conventional host-centered architectures.

Network 10 provides information on a wide variety of topics, including, but not limited to news, industry, financial needs, hobbies and cultural interests. Network 10 thus eliminates the need to consult multiple information sources, giving users an efficient and timesaving overview of subjects that interest them.

The transactional features of interactive network 10 saves the user time, money, and frustration by reducing time spent traveling, standing in line, and communicating with sales personnel. The user may, through RS 400, bank, send and receive messages, review advertisements, place orders for merchandise, and perform other transactions. In the preferred embodiment, network 10 provides information and transaction processing services for a large number of users simultaneously accessing the network via the public switched telephone network (PSTN), broadcast, and/or other media with their RS 400 units. Services available to the user include display of information such as movie reviews, the latest news, airlines reservations, the purchase of items such as retail merchandise and groceries, and quotes and buy/sell orders for stocks and bonds. Network 10 provides an environment in which a user, via RS 400 establishes a session with the network and accesses a large number of services. These services are specifically constructed applications which as noted are partitioned so they may be distributed without undo transmission time, and may be processed and selectively stored on a user's RS 400 unit.

SYSTEM CONFIGURATION

As shown in FIG. 1, in preferred form interactive computer network 10 includes four includes layers: information layer 100, switch and file server layer 200, 5 concentrator layer 300, and reception layer 401.

Information layer 100 handles: (1) the production, storage and dissemination of data and (2) the collection and off-line processing of such data from each RS session with the network 10 so as to permit the targeting of information to be presented to users and for traditional business support.

Switch and file server layer 200 and cache/concentrator layer 300 together constitute a delivery system 20 which delivers requested data to the RS 400's of reception layer 401 and routes data entered by the user or collected at RS 400's to the proper application in network 10. With reference to FIG. 2, the information used in a RS 400 either resides locally at the RS 400, or is available on demand from the cache/concentrator 300 20 or the file server 205, via the gateway 210, which may be coupled to external providers, or is available from information layer 100.

There are two types of information in the network 10 which are utilized by the RS 400: objects and messages. 25

Objects include the information requested and utilized by the RS 400 to permit a user to select specific parts of applications, control the flow of information relating to the applications, and to supply information to the network. Objects are self-describing structures organized in accordance with a specific data object architecture, described below. Objects are used to package presentation data and program instructions required to support the partitioned applications of a RS 400. Objects are distributed on demand throughout interactive network 10. Objects may contain: control information; program instruction to set up an application processing environment and to process user or network created events; information about what is to be displayed and how it is to be displayed; references to programs to be interpretively executed; and references to other objects, which may be called based upon certain conditions or the occurrence of certain events at the user's personal computer, resulting in the selection and retrieval of other partitioned applications packaged as 45 objects.

Messages are information provided by the user or the network and are used in fields defined within the constructs of an object, and are seen on the user's RS monitor 412, or are used for data processing at RS 400. Additionally, and as more fully described hereafter, messages are the primary means for communication within and without the network. The format of messages is application dependent. If the message is input by the user, it is formatted by the partitioned application currently being 55 processed on RS 400. Likewise, and with reference to FIG. 2, if the data are provided from a co-application database residing in delivery system 20, or accessed via gateway 210 or high function system 110 within the information layer 100, the partitioned application currently being processed on RS 400 causes the message 60 data to be displayed in fields on the user's display monitor as defined by the particular partitioned application.

All active objects reside in file server 205. Inactive objects or objects in preparation reside in producer system 120. Objects recently introduced into delivery system 20 from the producer system 120 will be available from file server 205, but may not be available on

cache/concentrator 302 to which the user's RS 400 has dialed. If such objects are requested by the RS 400, the cache/concentrator 302 automatically requests the object from file server 205. The requested object is routed back to the requesting cache/concentrator 302, which automatically routes it to the communications line on which the request was originally made, from which it is received by the RS 400.

The RS 400 is the point of application session control because it has the ability to select and randomly access objects representing all or part of partitioned applications and their data. RS 400 processes objects according to information contained therein and events created by the user on personal computer 405.

Applications on network 10 act in concert with the distributed partitioned applications running on RS 400. Partitioned applications are constructed as groups of objects and are distributed on demand to a user's RS 400. An application partition represents the minimum amount of information and program logic needed to present a page or window, i.e. portion of a page presented to the user, perform transactions with the interactive network 10, and perform traditional data processing operations, as required, including selecting another partitioned application to be processed upon a user generated completion event for the current partitioned application.

Objects representing all or part of partitioned applications may be stored in a user's RS 400 if the objects meet certain criteria, such as being non-volatile, non-critical to network integrity, or if they are critical to ensuring reasonable response time. Such objects are either provided on diskettes 426 together with RS 400 system software used during the installation procedure or, they are automatically requested by RS 400 when the user makes selections requiring objects not present in RS 400. In the latter case, RS 400 requests from cache/concentrator layer 300 only the objects necessary to execute the desired partitioned application.

Reception system application software 426 in preferred form is provided for IBM and IBM-compatible brands of personal computers 405, and all partitioned applications are constructed according to a single architecture which each such RS 400 supports. With reference to FIG. 2, to access network 10, a user preferably has a personal computer 405 with at least 512K RAM and a single disk drive 416. The user typically accesses network 10 using a 1,200 or 2,400 bps modem (not shown). To initiate a session with network 10, objects representing the logon application are retrieved from the user's personal diskette, including the R. S. application software, which was previously set up during a standard installation enrollment procedure with network 10. Once communication between RS 400 and cache/concentrator layer 300 has been established, the user begins a standard logon procedure by inputting a personal entry code. Once the logon procedure is complete, the user can begin to access various desired services (i.e., partitioned applications) which provide display of requested information and/or transaction operations.

APPLICATIONS AND PAGES

Applications, i.e. information events, are composed of a sequence of one or more pages opened at screen 414 of monitor 412. This is better seen with reference to FIG. 3a and 3b where a page 255 is illustrated as might appear at screen 414 of monitor 412. With reference to

5,347,632

9

FIG. 3a, each page 255 is formatted into page partitions 250, 260, 280, and 290 (not to be confused with applications partitions). Window page partitions 275, well known in the art, are also available and are opened and closed conditionally on page 255 upon the occurrence of an event specified in the application being run. Each page partition 250-290 and window 275 is made up of a page element which define the content of the partition or window.

Each page 255 includes: a header page partition 250, which has a page element associated with it and which typically conveys information on the page's topic or sponsor; one or more body page partitions 260 and window page partitions 275, each of which is associated with a page element which as noted gives the informational and transactional content of the page. For example, a page element may contain presentation data selected as a menu option in the previous page, and/or may contain prompts to which a user responds in predefined fields to execute transactions. As illustrated in FIG. 3b, the page element associated with body page partition 260 includes display fields 270, 271, 272. A window page partition 275 seen in FIG. 3a represents the same informational and transactional capability as a body partition, except greater flexibility is provided for its location and size.

Continuing with reference to FIG. 3b, advertisements 280 provided over network 10, like page elements, also include information for display on page 255, and may be included in any partition of a page. Advertisements 280 may be presented to the user on an individual basis from queues of advertisements that are constructed off-line by business system 130, and sent to file server 205 where they are accessible to each RS 400.

Individual queues of advertisements are constructed based upon data collected on the partitioned applications that were accessed by a user, and upon events the user generated in response to applications. The data are collected and reported by RS 400 to a data collection co-application in file server 205 for later transmission to business system 130. In addition to application access and use characteristics, a variety of other parameters, such as user demographics or postal ZIP code, may be used as targeting criteria. From such data, queues of advertisements are constructed and targeted to either individual users or to sets of users who fall into certain groups according such parameters.

Also with reference to FIG. 3b, a user interface 285 is displayed on the page which enables the user to interact with the network RS 400 and other elements of network 10, so as to cause such operations as navigating from page to page, performing a transaction, or obtaining more information about other applications. As shown in FIG. 3b, user interface 285 includes a command bar 290 having a number of commands 291-298 which the user can execute. The functions of commands 291-298 are discussed in greater detail below.

NETWORK OBJECTS

As noted above, in conventional time-sharing computer networks, the data and program instructions necessary to support user sessions are maintained at a central host computer. However, that approach has been found to create processing bottlenecks as greater numbers of users are connected to the network; bottlenecks which require increases in processing power and complexity; e.g., multiple hosts of greater computing capability, if the network is to meet demand. Further, such

10

bottlenecks have been found to also slow response time as more users are connected to the network and seek to have their requests for data processing answered.

The consequences of the host processing bottlenecking is to either compel capital expenditures to expand host processing capability, or accept longer response times; i.e., a slower network, and risk user dissatisfaction.

However, even in the case where additional computing power is added, and where response time is allowed to increase, eventually the host becomes user saturated as more and more users are sought to be served by the network. The method and apparatus of this invention are directed at alleviating the effects of host-centered limitations, and extending the network saturation point. In accordance with the invention, this is achieved by reducing the demand on the host for processing resources by structuring the network so that the higher network levels act primarily to maintain and supply data and programs to the lower levels of the network, particularly RS 400, which acts to manage and sustain the user screen displays.

More particularly, the method aspect of the invention features procedures for parsing the network data and program instructions required to support the interactive user sessions into packets, referred to as objects, and distributing them into the network where they can be processed at lower levels, particularly, reception system 400.

In accordance with the invention, the screens presented at the user's monitor are each divided into addressable partitions shown in FIG. 3a, and the display text and graphics necessary to make up the partitions, as well as the program instructions and control data necessary to deliver and sustain the screens and partitions are formulated from pre-created objects. Further, the objects are structured in accordance with an architecture that permits the displayed data to be relocatable on the screen, and to be reusable to make up other screens and other sessions, either as pre-created and stored sessions or interactive sessions, dynamically created in response to the user's requests.

In accordance with the method aspect of the invention and as shown in FIG. 4c, the network objects are organized as a family of objects each of which perform a specific function in support of the interactive session. More particularly, the network object family is seen to include 6 members: page format objects 502, page element object 504, window objects 506, program objects 508, advertisement objects 510 and page template objects 500.

Within this family, page format objects 502 are designed to define the partitioning 250 to 290 of the monitor screen shown in FIG. 3a. The page format objects 502 provide a means for pre-defining screen partitions and for ensuring a uniform look to the page presented on the reception system monitor. They provide the origin; i.e., drawing points, and dimensions of each page partition and different values for presentation commands such as palette and background color.

Page format objects 502 are referenced whenever non-window data is to be displayed and as noted ensure a consistent presentation of the page. In addition, page format objects 502 assures proper tessellation or "tiling" of the displayed partitions.

Page element objects 504, on the other hand, are structured to contain the display data; i.e., text and graphic, to be displayed which is mapped within screen

5,347,632

11

partitions 250 to 290, and to further provide the associated control data and programs. More specifically, the display data is described within the object as NAPLPS data, and includes, PDI, ASCII, Incremental Point and other display encoding schemes. Page element objects also control the functionality within the screen partition by means of field definition segments 516 and program call segments 532, as further described in connection with the description of such segments hereafter. Page element objects 504 are relocatable and may be reused by many pages. To enable the displayable data to be relocated, display data must be created by producers in the NAPLPS relative mode.

Continuing with reference to FIG. 4c, window objects 506 include the display and control data necessary to support window partitions 275 best seen in FIG. 3a. Windows contain display data which overlays the base page and control data which supersede the base page control data for the underlying screen during the duration of the window. Window objects 506 contain data which is to be displayed or otherwise presented to the viewer which is relatively independent from the rest of the page. Display data within windows overlays the base page until the window is closed. Logic associated with the window supersedes base page logic for the duration of the window. When a window is opened, the bitmap of the area covered by window is saved and most logic functions for the overlaid page are deactivated. When the window is closed, the saved bit map is swapped onto the screen, the logic functions associated with the window are disabled, and prior logic functions are reactivated.

Windows are opened by user or program control. They do not form part of the base page. Windows would typically be opened as a result of the completion of events specified in program call segments 532.

Window objects 506 are very similar in structure to page element objects 504. The critical difference is that window objects 506 specify their own size and absolute screen location by means of a partition definition segment 528.

Program object 508 contain program instructions written in a high-level language called TRINTEX Basic Object Language, i.e., TBOL, described in greater detail hereafter, which may be executed on reception system 400 to support the application. More particularly, program objects 508 includes interpretable program code, executable machine code and parameters to be acted upon in conjunction with the presentation of text and graphics to the reception system monitors.

Program objects 508 may be called for execution by means of program call segments 532, which specify when a program is to be executed (event), what program to execute (program pointer), and how programs should run (parameters).

Programs are treated as objects to conform to the open-ended design philosophy of the data object architecture (DOA), allowing the dissemination of newly developed programs to be easily and economically performed. As noted above, it is desirable to have as many of these program objects staged for execution at or as close to RS 400 as possible.

Still further, advertising objects 510 include the text and graphics that may be presented at ad partition 280 presented on the monitor screen as shown in FIG. 3b.

Finally, the object family includes page template objects 500. Page template objects 500 are designed to

12

define the components of the full screen presented to the viewer. Particularly, page template objects 500 include the entry point to a screen, the name of the page format objects which specify the various partitions a screen will have and the page element object that contain the display data and partitioning parameters for the page.

Additionally, page template object 500 includes the specific program calls required to execute the screens associated with the application being presented to the user, and may serve as the means for the user to selectively move through; i.e., navigate the pages of interest which are associated with various applications. Thus, in effect, page template objects 500 constitute the "recipe" for making up the collection of text and graphic information required to make the screens to be presented to the user.

Also in accordance with the invention, object 500 to 510 shown in FIG. 4c are themselves made up of further sub-blocks of information that may be selectively collected to define the objects and resulting pages that ultimately constitute the application presented to the user in an interactive text and graphic session.

More specifically and as shown schematically in FIG. 4a, objects 500 to 510 are predefined, variable length records consisting of a fixed length header 551 and one or more self-defining record segments 552 a list of which is presented in FIG. 4c as segment types 512 to 540.

In accordance with the invention, and as shown in FIG. 4b, object header 551 in preferred form is 18 bytes in length and contains a prescribed sequence of information which provides data regarding the object's identification, its anticipated use, association to other objects, its length and its version and currency.

More particularly, each of the 18 bytes of object header 551 are conventional hexadecimal, 8 bit bytes and are arranged in a fix pattern to facilitate interpretation by network 10. Particularly, and as shown in FIG. 4b, the first byte of header 551; i.e., byte 1, identifies the length of the object ID in hexadecimal. The next six bytes; i.e., bytes 2 to 7, are allocated for identifying access control to the object so as to allow creation of closed user groups to whom the object(s) is to be provided. As will be appreciated by those skilled in the art, the ability to earmark objects in anticipation of user requests enables the network anticipate requests and pre-collect objects from large numbers of them maintained to render the network more efficient and reduce response time. The following 4 bytes of header 551; bytes 8 to 11, are used to identify the set of objects to which the subject object belongs. In this regard, it will be appreciated that, again, for speed of access and efficiency of selection, the objects are arranged in groups or sets which are likely to be presented to user sequentially in presenting the page sets; i.e., screens that go to make up a session.

Following identification of the object set, the next byte in header 551; i.e., byte 12, gives the location of the subject object in the set. As will be appreciated here also the identification is provided to facilitates ease of object location and access among the many thousands of objects that are maintained to, thereby, render their selection and presentation more efficient and speedy.

Thereafter, the following bytes of header 551; i.e., byte 13, designates the object type; e.g., page format, page template, page element, etc. Following identification of the object type, two bytes; i.e., bytes 14, 15, are

5,347,632

13

allocated to define the length of the object, which may be of what ever length is necessary to supply the data necessary, and thereby provides great flexibility for creation of the screens. Thereafter, a single byte; i.e., byte 16, is allocated to identify the storage characteristic for the object; i.e., the criterion which establishes at what level in network 10 the object will be stored, and the basis upon which it will be updated. At least a portion of this byte; i.e., the higher order nibble (first 4 bits reading from left to right) is associated with the last byte; i.e., byte 18, in the header which identifies the version of the object, a control used in determining how often in a predetermined period of time the object will be updated by the network.

Following storage characteristic byte 16, header 551 includes a byte; i.e., 17, which identifies the number of objects in the set to which the subject object belongs. Finally, and as noted above, header 551 includes a byte; i.e., 18, which identifies the version of the object. Particularly the object version is a number to establish the control for the update of the object that are resident at reception system 400.

As shown in FIG. 4a, and as noted above, in addition to header 551, the object includes one more of the various segment types shown in FIG. 4c.

Segments 512 to 540 are the basic building blocks of the objects. And, as in the case of the object, the segments are also self-defining. As will be appreciated by those skilled in the art, by making the segments self-defining, changes in the objects and their use in the network can be made without changing pre-existing objects.

As in the case of objects, the segments have also been provided with a specific structure. Particularly, and as shown in FIG. 4a, segments 552 consists of a designation of segment type 553, identification of segment length 554, followed by the information necessary to implement the segment and its associated object 555; e.g., either, control data, display data or program code.

In this structure, segment type 553 is identified with a one-byte hexadecimal code which describes the general function of the segment. Thereafter, segment length 554 is identified as a fixed two-byte long field which carries the segment length as a hexadecimal number in INTEL format; i.e., least significant byte first. Finally, data within segments may be identified either by position or keyword, depending on the specific requirements of the segment.

In accordance with the invention, the specific structure for the objects and segments shown in FIG. 4c would be as described below. In that description the following notation convention is used:

< >	- mandatory item
()	- optional item
...	- item may be repeated
item item	
< > ()	- items in a column indicate either/or
item item	

The structure for objects is:

PAGE TEMPLATE OBJECT,

[<header> (compression descriptor) <page format call> (page element call) ... (program call) ... (page element selector) (system table call) ... external reference) (keyword/navigation)];

14

As noted above, page format objects 502 are designed to define the partitioning 250 to 290 of monitor screen 414 shown in FIG. 3a.

PAGE FORMAT OBJECT,

[<header> (compression descriptor) (page defaults) <partition definition>];

PAGE ELEMENT OBJECT,

[<header> (compression descriptor) (presentation data) ... (program call) ... (custom cursor) ... (custom text) ... (field definition) ... (field-level program call) ... (custom cursor type 2) ... (custom graphic) ... (field definition type 2) ... (array definition) ... (inventory control)];

Page element objects, as explained, are structured to contain the display data; i. e. , text and graphics, to be presented at screen partitions 250 to 290.

WINDOW OBJECT,

[<header> (compression description) <partition definition> (page element call) (presentation data) ... (program call) ... (custom cursor) ... (custom text) ... (custom cursor type 2) ... (custom graphic) ... (field definition) ... (field level program call) ... (field definition type 2) ... (array definition) ... (inventory control)];

As noted, window objects include display and control data necessary to support window partition at screen 414 .

PROGRAM OBJECTS,

[<header> (compression descriptor) <program data> ...].

Program objects, on the other hand, contain program instructions written in higher-level language which may be executed at RS 400 to support the application.

ADVERTISEMENT OBJECT,

[<header> (compression descriptor) (presentation data) ... (program call) ... (custom cursor) ... (custom text) ... (field definition) ... (field-level program call) ... (custom cursor type 2) ... (custom graphic) ... (field definition type 2) ... (array definition) ... (inventory control)];

As can be seen, advertisement objects are substantially the same as page element objects, with the difference being that, as their name implies, their subject matter is selected to concern advertising.

Continuing, in accordance with the invention, the structure for the object segments is as described hereafter.

PROGRAM CALL SEGMENT

Program call segments 532 are used to invoke programs. Program events will be specified in logical terms and will be mapped by the reception system native software 420 to specific physical triggers (e.g., the "logical" event end of page may map to the physical <ENTER> key). The logical event to be completed to initiate the program is specified in a one-byte token within the segment. The structure of program call segment 532 is as follows:

[<st> <sl> <event> <prefix> <	prgm obj. id
	> (parm)];
	displacement

where "st" is type; "sl" length; "event" is a one-byte token of the logical event to be completed to initiate the program; "prefix" is a one-byte prefix to an object id or displacement; "object id" is id of the program object 508; "displacement" is a pointer to an imbedded pro-

5,347,632

15

gram call segment 532; and "parm" is the parameters specific to the program.

FIELD LEVEL PROGRAM CALL SEGMENTS

Some programs, such as edits, must be triggered at the field level. Field-level program call segments 518 relate program calls to specified field definition segments 516. The structure of field-level program call segments is as follows:

```
[<st> <sl> <event> <field id> <prefix> <
| prgm.obj.id |
> (parm) . . . ;
|displacement|
```

where "st" is type; "sl" length; "event" is a one-byte token of the logical event to be completed to initiate the program; "field id" is the one-byte name of the field specified in a field definition segment 516 with which this call segment is associated; "prefix" is a one-byte prefix to an object id or displacement; "object id" is id of the program object 506; "displacement" is a pointer to an imbedded program call segment 532; and "parm" is the parameters specific to the program.

PROGRAM DATA SEGMENT

Program data segments 536 contain the actual program data to be processed by RS 400. Program data may include either source code, compiled machine code, macros, storage maps, and/or parameters. The structure of program data segments 536 is as follows:

```
[<st> <sl> <type> <program data>];
```

where "st" is type; "sl" length; "type" refers to the type of program data contained; i.e., (1=TBOL, 2=table data); and "program data" is the actual program to be executed.

COMPRESSION DESCRIPTOR SEGMENT

Compression descriptor segment contains information need for the decompression of objects compressed in interactive network 10. The segment is a formalization of parameters to be used by a decompression routine residing at the RS 400, using; for example, Huffman encoding well known the art. The structure of compression descriptor segment 513 is:

```
[<st> <sl> <table number> <length 1> (length 2)];
```

where "st" is type; "sl" length; "table number" is a one-byte number corresponding to the "class" indicator in the table structure segment of the appropriate decompression system table object; "length 1" is a two-byte indicator of the length of the segment after compression (not including object header and length of compression descriptor); and "length 2" is a two-byte indicator of the length of the segment before compression (not including object header and length of compression descriptor).

PAGE DEFAULT SEGMENTS

Page default segments 540 specify defaults for the entire page using NAPLPS commands. The structure of page default segment 540 is:

```
[<st> <sl> <NAPLPS>];
```

where "st" is type; "sl" length; and "NAPLPS" are the commands that may be used to specify default characteristics of the page.

PARTITION DEFINITION SEGMENT

16

Partition definition segment 528 describes display screen areas into which data may be mapped. The structure of partition definition segment 528 is:

```
[<st> <sl> <partition id> <origin> <size> (NAPLPS)];
```

where "st" is type; "sl" length; "partition id" is a one-byte partition id unique within the current page format object 502; "origin" is the partition origin point, a three-

byte NAPLPS point set (absolute, invisible) operand contained the absolute coordinates of the lower left corner of the partition; and "size" refers to partition size, a three-byte NAPLPS point set (absolute, invisible) operand containing the absolute coordinates of the upper right corner of the partition.

PAGE FORMAT CALL SEGMENT

Page format call segment 526 is used by the page template object 500 to specify the particular page format object 502 to be used as the "blueprint" of the page. Page format call segment 526 structure is as follows:

```
[<st> <sl> <prefix> <object id>];
```

where "st" is type; "sl" length; "prefix" is a one-byte prefix to an object id or displacement; and "object id" is the object id of the page format object 502.

PAGE ELEMENT CALL SEGMENT

Page element call segment 522 specifies which data is to be present on the base page and in which page partition the data is to appear. The structure of page element call segment is as follows:

```
[<st> <sl> <partition id> <priority> <prefix> <
| object id |
> ];
|displacement|
```

where "st" is type; "sl" length; "partition id" is the partition id, as specified in the page format object 502 upon which this object will act; "priority" is a one-byte binary flag indicating priority (from 0-15 with 0 indicating no priority [FIFO]) of object interpretation (high-order nibble) and of painting (low-order nibble); "prefix" is a one-byte object id or displacement; "object id" is the id of the page element object 504; and "displacement" is a pointer to an imbedded page element object 533.

PAGE ELEMENT SELECTOR SEGMENT

Page element selector segment 524 provides a mechanism by which page elements may be dynamically selected for presentation within a partition. The structure of page element selector segment 524 is:

```
[<st> <sl> <part.id> <priority> <prefix> <
| pgm.obj.id |
> (parm) . . . ;
|displacement|
```

where "st" is type; "sl" length; "part. id" is the partition id as specified within the page format object 502 upon which the object will act; "priority" is a one-byte binary flag indicating priority (from 0-15 with 0 indicating no priority [FIFO]) of object interpretation (high-order nibble) and of painting (low-order nibble); "prefix" is a one-byte object id or displacement; "pgm.obj.id" is the object id of the program object 508 used to dynamically select an element object; "displacement" is a pointer to

5,347,632

17

an imbedded program object 508, and "parre" is parameters which are used by the program object 508.

SYSTEM TABLE CALL SEGMENT

System table call segments 537 call system table segments for use by the RS 400. Each table entry in a system table segment contains an index-addressable segment (e.g., a set of custom text segments 514). System table call segments operate in a "locked-shift" mode, meaning that each system table of a particular class will remain operative until a new table is requested for that class of table. System table call segment 537 structure is as follows:

```
[<st> <sl> <prefix> <
    | object id |
    > ];
    |displacement|
```

where "st" is type; "sl" length; "prefix" is a one-byte prefix to an object id or displacement; "object id" is the id of a system table segment; and "displacement" is a pointer to an imbedded system table segment.

TABLE STRUCTURE SEGMENT

Table structure segments 531 describe the basic class and composition of system table objects. The structure of table structure segment 531 is:

```
[<st> <sl> <class> <number of entries>
<maximum entry length>];
```

where "st" is type; "sl" length; "class" is a one-byte identifier indicating the class of the current table (as follows:

```
x'00'=custom text table
x'01'=custom cursor table
x'02'=custom graphic table
x'03'=custom cursor type 2 table
x'30' thru x'39'=decompression table)
```

"number of entries" is a two-byte field specifying the total number of entries contained in the current table; and "maximum entry length" is a two-byte field specifying the length of the largest entry in the current table.

TABLE ENTRY SEGMENT

Table entry segment 535 contains the actual data that has been placed in tabular form. The meaning of the data is derived from the class indicator in the table structure segment 554. They will be treated as functional equivalent of certain other segments such as custom text segment 514 or custom cursor segment 512. Table entry segment structure is:

```
[<st> <sl> <data>];
```

where "st" is type; "sl" length; and "data" is the data contained in the entry (text character attributes if table belongs to the custom text class; NAPLPS if the table belongs to the custom cursor class).

EXTERNAL REFERENCE SEGMENT

External reference segment 523 is provided to improve run-time performance by providing the RS 400 with a list of objects that are candidates for pre-fetching. External reference segments 523 contain a list of object-ids which are used within the current page. Each object indicated within this list is called explicitly from the current frame. Object ids specified within the external reference segment 523 will take advantage of the notion of "inheritance." If multiple object ids are contained within the segment, they may inherit high-order bytes from previously specified ids, thus avoiding repetition of information that is inherited (e.g. to specify objects ABC12, ABC22, and ABC37 in this segment, one encodes them as ABC12, 22, 37). External reference segments 523 operate in a "locked-shift" mode, meaning that each external reference list will be active until the

18

next external reference list is encountered. In the best mode, there should be no more than one external reference segment per page. External reference segment structure is as follows:

```
[<st> <sl> <# of ids> <priority> <prefix>
<object id>];
```

where "st" is type; "sl" length; "# of ids" is a one-byte field specifying the total number of object ids contained in the current segment; "priority" is a one-byte priority value specifying priority of pre-fetch (priorities may be duplicated, in which case they will be processed from left to right); "prefix" is a one-byte prefix to an object id or displacement; and "object id" is the id of an externally referenced object.

KEYWORD/NAVIGATION SEGMENT

Keyword/navigation segments 520 may contain two types of information: (1) references to other page template objects 500 that are either logically higher than the current page template (e.g., a "parent" menu) or references to page template objects 500 outside the current "world" (a logically cohesive group of pages having a single entry point, such as a general map of the interactive service); or (2) a character string to be associated with the current page template object 500, which may be displayed to the user to indicate an alternative path or keyword which could be used to access the current page template. The structure of keyword/navigation segment is as follows:

```
[<st> <sl> <#ids> (<prefix> <object id>) . .
. (character string) ];
```

where "st" is type; "sl" length; "#ids" is the number of object ids in this segment; "pre-fix" is a one-byte object id prefix; "object id" is an object id associate with the current page as either an upward hierarchical reference or a non-hierarchical reference; and "character string" is the character string to be associated with the current page. (See also, discussion of Jump word navigation, below).

PRESENTATION DATA SEGMENT

Presentation data segments 530 contain the actual data to be displayed or otherwise presented to the user. Presentation data may contain NAPLPS codes, ASCII, and other codes for visual display. Presentation data may in the future contain codes for the presentation of audio signals. The structure of presentation data segment is:

```
[<st> <sl> <type> <size> <presentation
data>];
```

where "st" is type; "sl" length; "type" is the type of presentation data included in this segment (1=NAPLPS, 2=ASCII); "size" is a NAPLPS operand that defines the upper right portion of the display data; and "presentation data" is the actual data to be presented to the user.

FIELD DEFINITION SEGMENT

Field definition segments 516 define the location of a field, name the field, and specify how data will be acted on within the named field. Field definition segment 516 structure is as follows:

```
[<st> <sl> <attributes> <origin> <size>
<name> <text id>(cursor id) (cursor origin) ];
```

where "st" is type; "sl" length; and the structure is defined as below. "Attributes" of a field define ways in which the user interacts with RS 400 at a rudimentary level. Three basic field types are supported: (1) unprotected fields into which users may enter data; (2) protected fields into which users may position the cursor,

function and enter keys, but may not enter data; and (3) skip fields which are inaccessible to the user keyboard. Additional attributes which may be specified for a field include: numeric input only (unprotected); alphabetic input only (unprotected); foreground color; and background color. Attributes are encoded in two bytes. The first nibble of the first byte is a hexadecimal number (O-F) that represents the foreground color selection from the in-use palette. The second nibble of the first byte is a hexadecimal number (O-F) that represents the background color selection from the in-use palette. The first nibble of the second byte consists of a set of bit flags which, from left to right, indicate:

- bit 0 if '1': protect on;
- bit 1 if '1': automatic skip on;
- bit 2 if '1': numeric input only; and
- bit 3 if '1': alphabetic input only.

The second nibble of the second byte is reserved to accommodate for expansion of network 10.

Continuing, "Origin" is a three-byte NAPLPS point set (relative, invisible) operand that defines the lower left corner of the field; "Size" is a three-byte NAPLPS point set (relative, invisible) operand that defines the upper right corner of the field; "Name" is a one-byte name assigned to the field so that it may be accessible to programs; "Text id" is a one-byte id of the text characteristics to be associated with the field (e.g., size, gapping, proportional spacing, etc.); "Cursor id" is a one-byte id of the cursor type to be associated with the field; "Cursor origin" is a three-byte NAPLPS operand specifying relative draw point to the cursor, if this operand is not present, the cursor origin point will be assumed to be the same as the field origin point.

FIELD DEFINITION TYPE 2 SEGMENT

Field definition type 2 segments 517 are provided to enhance runtime flexibility of fields. Field definition type 2 segment structure is as follows:

[<st> <sl> <attributes> <origin> <size> <name> <text id> <cc 11>(<cursor id>(cursor origin)) <# hot spots>(<hs 11> <hssize>(horigin))...(<cg 11> <cgraphic id> <cgmcode>(cgorigin))...];

where structure is defined below. As with the other segments, "st" describes segment type, and "sl" segment length. Further, "Attributes" describe how the user and RS 400 interact at a rudimentary level. Attributes for field definition type 2 segments 517 are contained in four bytes:

Byte 1	Field type
bit 0	TBOL interpreter indicator: no fire; or fire
bits 1-7	Interaction type input (unprotected); action (protected); display (askip); and hidden (dark)
Byte 2	Text Attributes (bit flags)
bits 0-7	left justify; right justify; and word wrap
Byte 3	Data Type:
bits 0-7	alphabetic; numeric; password;
Byte 4	Color:
bits 0-3	foreground color;
bits 4-7	background color.

"Origin" is a three-byte NAPLPS point set (relative, invisible) operand that defines the lower left corner of the field. "Size" is a three-byte NAPLPS point set (relative, invisible) operand that defines the upper right corner of the field. "Name" is a one-byte name assigned to the field so that it maybe accessible to the program. "Text id" is a one-byte id of the text characteristics to be associated with the field, such as size, gapping, proportional, etc. "cc 11" is the cursor length; a one-byte field describing the combined length of the cursor id field and the cursor origin field. If the length contains a 1, then the cursor origin operand is not present, in which case, the cursor origin defaults to the field origin point. "Cursor id" is a one-byte id of the cursor type to be associated with the field. "Cursor origin" is a three-byte NAPLPS operand specifying the relative draw point of the cursor. If this operand is not present, the cursor origin point will be assumed to be the same as the field origin point "# hot spots" is the number of hot spots used by this field. "Hot spots" refers to a set of coordinates that will be selectable by a pointing device, such as a mouse. If the contents of this field are zero, the hot spot for the field will be assumed to be the coordinates that are covered by the custom cursor. "Hot spot sets" facilitate assigning a variable number of hot spots to a field. Each hot spot is described by a set of operand consisting of hot spot length, origin, and size. Each set of such operand describes one hot spot. When using multiple hot spots, multiple sets of operand must be present. "hs 11" or hot spot length is a one-byte binary field describing the length of the hot spot coordinates for a hot spot "instance." If this byte contains zero, the hot spot origin and size default to the coordinates described by the custom cursor. If this byte contains 3, then the hot spot origin point will not follow, but will default to the custom cursor origin point. If this byte contains 6, then both the hot spot origin and size are present. "Hot spot size" is a three-byte NAPLPS x,y coordinate describing the top right corner of the hot spot. "Hot spot origin" is a three-byte NAPLPS x,y coordinate describing the lower left corner of the hot spot. If the hot spot length is equal to 3, this field is not present. In that case, the hot spot origin point defaults to the origin point of the custom cursor (which may have also defaulted to the field origin point). If the hot spot length is equal to 6, then this field is present. A custom graphic operand set contains four operand each of which is given in the Field Definition Segment as shown. Particularly: "cg 11" is the custom graphic set length, which, if 2, then no custom graphic origin is present. In that case, the origin point of the custom graphic defaults to the field origin point; "cg id" is the custom graphic id, a one-byte identifier of a custom graphic string; "cgmcode" is the custom graphic mode, which is one byte used to describe variable conditions that apply to the graphic. Defined values include: x'01: blink; x'02: dynamic; x'03: permanent; and "cgorigin" is the custom graphic origin, a three-byte NAPLPS x,y coordinate indicating the lower left corner of the custom graphic. If this operand is not present, the lower left corner will default to the field origin point.

ARRAY DEFINITION SEGMENT

Array definition segments 515 define the names and relative locations of fields in a row that makes up an array or table. The first row of fields must have been defined using field definition segments 516. The array definition provides a short hand for specifying the repli-

cation of selected fields from the initial page. The structure of the array definition segment 515 is as follows:

[<st> <sl> <# occurrences> <vertical gap> <field name> ...];
where "st" is type; "sl" length; "# occurrences" is a one-byte field describing the number of rows to be generated to create the array (the first row is assumed to be generated from field definition segments 516); "vertical gap" is a NAPLPS point set operand (relative, invisible) containing the DY of inter-row spacing; and "field name" is a one-byte name (from the field definition) of the fields in a row of the array.

CUSTOM GRAPHICS SEGMENT

Custom graphics segment 521 provides a means to package graphics commands. These graphics commands may be related to a field and initiated based on run-time conditions. The structure of custom graphics segment 550 is as follows:

[<st> <sl> <id> <size> <NAPLPS>];
where "st" is type; "sl" length; "id" is a one-byte identifier for this custom graphic; "size" is a three-byte NAPLPS operand specifying upper right corner of the graphic area in a relative mode; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM CURSOR SEGMENT

Custom cursor segment 512 allows fancy graphics to be associated with cursor positioning in a field. Using this segment, cursor may be defined to any size or shape and may be placed at any desired location relative to their associated fields. The structure of custom cursor segment 512 is as follows:

[<st> <sl> <id> <size> <NAPLPS>];
where "st" is type; "sl" length; "id" is a one-byte identifier for this custom cursor; "size" is a three-byte NAPLPS operand specifying upper right corner of the cursor area in a relative mode; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM CURSOR TYPE 2

Custom cursor type 2 segment 519 allows cursor to be defined to any size or shape and may be placed at any desired location relative to their associated fields. The structure of custom cursor type 2 segment 519 is as follows:

[<st> <sl> <id> <size>(<11> <NAPLPS>) ...];
where "st" is type; "sl" length; "id" is a one-byte identifier for this custom cursor; "size" is a three-byte NAPLPS operand specifying upper right corner of the cursor area in a relative mode; "11" is the length of the following NAPLPS data; and "NAPLPS" are NAPLPS commands to paint the custom image.

CUSTOM TEXT SEGMENT

Custom text segments 514 allow the definition of custom display of text within a field when non-standard character field size is used (20×40 display characters is standard) or custom spacing, movement, or rotation of characters is desired. The structure of custom text segments 514 is as follows:

<st> <sl> <id> <NAPLPS>;
where "st" is type; "sl" length; "id" is a one-byte identifier for this TXT command; and "NAPLPS" are NAPLPS commands specifying character field size, rotation, movement, inter-row and inter-character text gaps.

INVENTORY CONTROL SEGMENT

Inventory control segment 535 is provided to facilitate management of objects. The inventory segment is structured:

[<st> <sl> <type> <inventory number>(sub-number)];

where "st" is type; "sl" length; "type" is a one-byte indicator showing object usage as follows: 0=no defined use; 1=leader ad; 2=ad campaign completion; 3=leader ad completion; 4-255=reserved for future use); "inventory number" is a unique two-byte number to be used for inventory control and statistics; and "sub-number is the same as inventory number.

As shown in FIG. 4C, the family of object segments also includes imbedded objects and elements; i.e., segments 533 and 525, which represent objects and elements nested; i.e., imbedded within objects. As will be appreciated, the formulation of imbedded objects and elements would be as described above for objects and elements generally and, further, would be consistent with the described structure for segments.

NETWORK MESSAGES

In addition to the network objects, and the display data, control data, and the program instructions they contain as previously described, network 10 also exchanges information regarding the support of user sessions and the maintenance of the network as "messenger". Specifically, messages typically relate to the exchange of information associated with initial logon of a reception system 400 to network 10, dialogue between RS 400 and other elements and communications by the other network elements amongst themselves.

In accordance with the invention, to facilitate message exchange internally, and through gateway 210 to entities externally to network 10, a protocol termed the "Data Interchange Architecture" (DIA) is used to support the transport and interpretation of information. More particularly, DIA enables: communications between RS 400 units, separation of functions between network layers 100, 200, 300 and 401; consistent parsing of data; an "open" architecture for network 10; downward compatibility within the network; compatibility with standard industry protocols such as the IBM System Network Architecture; Open Systems Interconnections standard; support of network utility sessions; and standardization of common network and application return codes.

Thus DIA binds the various components of network 10 into a coherent entity by providing a common data stream for communications management purposes. DIA provides the ability to route messages between applications based in IBM System Network Architecture (SNA), (well known in the art, and more fully described in *Data and Computer Communications*, by W. Stallings, Chapter 12, McMillian Publishing, Inc. (1985)) and non-SNA reception system applications; e.g. home computer applications. Further, DIA provides common data structure between applications run at RS 400 units and applications that may be run on external computer networks; e.g. Dow Jones Services, accessed through gateway 210. As well, DIA provides support for utility sessions between backbone applications run within network 10 as described hereafter.

In make up, DIA is a blend of SNA and non-SNA based modes, and thus provides a means for combining the differences between these modes within network 10. Accordingly, the action of DIA differs depending on whether DIA is operating within an SNA portion of network 10 or whether it is operating within the non-SNA portion of the network. More specifically, within the SNA portion of network 10, DIA and its supporting

5,347,632

23

programs may be considered "applications" facilities. In this context, DIA resides at the transaction services level of SNA, also known as the Specific Application level of Open Systems Interconnections (OSI, also discussed in chapter 12 of *Data and Computer Communications* by W. Stallings above noted). However, in either case, it is a level 7 facility.

Within non-SNA portions of network 10, DIA and its supporting programs provide routing, transport, sessions, and some transaction facilities. Thus DIA provides a comprehensive network architecture providing OSI level 3, 4, 5 and 7 services.

In accordance with the invention, DIA facilitates "utility session" within network 10. Utility sessions allow partner applications to communicate by means of the single session established between two logical units of the SNA type. In order to reduce the number of resources which must be defined to the network support programs, many user messages may be passed to many different application destinations through logical unit to logical unit (LU-LU) "pipes".

Applications exist on either side of the LU-LU pipe which act to concentrate outbound messages en route to applications resident on the other side of the LU-LU pipe; distribute inbound messages to local applications; and maintain and manage application task boundaries. Users may enter into a conversation with a set of transactions, refined to tasks, which are hereafter noted as "user sessions", and the boundaries of these user sessions (tasks) are indicated by begin session/end session flags.

Another application function supported by DIA is the routing of messages between nodes of network 10. Particularly, a switching application will route messages to the appropriate LU-LU session for transmission to another node by examining and resolving the DIA destination IDs hereafter described.

In accordance with the invention messages conforming to DIA are composed of two functional parts: message headers and message text. Message Headers are transparent to most applications, but are the primary vehicle for passing information for session layer to session layer or transport layer to transport layer communications. Further, Message Text which is processed by end users, and is transparent to session and transport mechanisms.

In order to reduce program complexity and facilitate maintenance and enhancements, DIA has been structured in a layered fashion. In this regard, the DIA-defined data which flows through network 10 consists of a set of headers preface the end-user to end-user message text. Further, as in the case of objects, messages are organized in a family of types based on the specific form of its header. Particularly, there are "FMO" headers which contain routing and control information; FM2 headers which contain transport level information; FM4 headers which contain gateway information; FM8 headers which obtain information for secondary routing; i.e. messages passed through from node to node; FM9 headers which contain network management information; and FM64 headers which contain application-to-application management information, where, for example, applications running at RS 400 need be rendered compatible with applications running on an external computer connected to network 10 through a gateway 210.

In order to provide SNA compatibility, the first two bytes of all DIA FM headers are formatted such that

24

byte 1 defines the length of header in hexadecimal; and byte 2, bit 0, identifies whether concatenation is provided or not; e.g. if bit 1=0 no other headers follow, but if bit 1=1, then the current header is followed by a concatenated header; while bits 1-7 identify the header type in hexadecimal value.

As will be appreciated to those skilled in the art, this layout is the same as that of SNA Function Management Headers. In an SNA LU0 implementation the DIA FM headers may be treated as SNA Function Management Headers (FMHs). Alternatively, the DIA FMHs may be treated as pure data within the SNA Request Unit (RU).

With regard to destination routing, the basic premise of DIA is that each message flowing through network 10 carries a DIA header (FM0) that identifies its source and destination ids. Accordingly, switching applications exist which map destination ids to resources and route messages appropriately. In accordance with the invention, in order to send a reply, the recipient application simply swaps the content of the destination and source id fields and return message.

In the context of DIA the totality of ports, devices, and programs which are managed by a particular Switch and defined as destinations, are referred to as "regions" In this regard, each Switch; i.e. server 205 or cache/concentrator 302 shown in FIG. 2, need only be aware of the destination ids of resources within its own region and of the destination ids of switches resident in immediately adjacent nodes. Since server 205 is the central hub within the network 10 for application message routing, messages destined for end-users unknown to a switch are routed toward server 205 for eventual resolution. Destination id naming conventions then enable server 205 to determine the appropriate switch to which the message should be forwarded. Particularly, "destination id" fields "regions" and "unit" are used for this purpose.

Concerning switch responsibility, a switching application has three primary responsibilities. It must forward messages to adjacent switches. It must collect messages from, and distribute messages to resources within its own region. And, it must maintain and manage application task boundaries. Users may enter into a conversation with a set of transactions. This set of transactions is referred to as a "task". These tasks are called user sessions. Further, the boundaries of these tasks are indicated by begin session/end session flags.

In order to fulfill these functions, a resource definition facility must exist for each switch to map each addressable resource to a destination id. In some cases, particularly on the RS 400, it may be desirable for an application to dynamically define subordinate resources to the switch and to interact with the switch to generate unique destination ids for these subordinate resources. It may also be necessary for the switch to either communicate with, or act within an application subsystem. An example of an application subsystem is the Customer Information Control System, (CICS) event, where CICS is a commercially available transaction process controller of the IBM Company, well known in the art. CICS, although subordinate to the operating system, is responsible for initiating and managing application "transaction" programs. Routing to specific transactions under the control of an application subsystem may be accomplished by a secondary address. In this case, the subsystem is defined as the primary destination. The transaction is defined as the secondary destination. A

25

switch must only route incoming messages to the subsystem. The subsystem in turn posts to, or initiates the desired transaction.

The use of secondary addressing provides several advantages. Particularly, switch resource tables are not affected by the coming and going of "transaction" applications. Further, since the DIA headers are SNA compatible, Type 1 application such as CICS need have no special message routing functions. A switch configured in accordance with the IBM standard VTAM could route incoming messages to CICS. Still further, transactions need not go into "receive loops". It is possible for the subsystem to poll on behalf of many transaction programs. In accordance with DIA, secondary addressing is implemented within the application data stream. For instance, CICS transaction ids are, by convention, to be found in the first four bytes of application text.

With regard to the standards for DIA, it will be recalled that DIA messages have a header followed by the message information. In the preferred embodiment, the DIA headers may be concatenated to one another. Further, the presence of concatenated headers is indicated by the setting of the first bit (bit 0) of the Header Type field.

However, there are two restrictions on the use of concatenated headers. Particularly, concatenated headers are required to be sequenced in ascending order left to right by header type numbers and secondary message text prefaced by concatenated headers (such as FM64 architecture message text) are not permitted to span across message block.

The basic structure of all DIA headers is presented below. As presented, "< >" indicate mandatory elements, "(" indicate optional elements and "." indicate repeat allowed. Further, the "FMX" designations refer to the message header types previously identified and "TTX" denotes TRINTEX, the former name of the network developer.

The basic DIA header structure is:

[<Length> <Concatenation flag <Type>(FM defined data)].

For TTX application-to-application messages, the structure is:

[(FM0) (FM2) (FM8) (<FM64> (64text)) . . . (Appl Text)]

For TTX application-to-gateway application messages, the structure is:

[(FM0) (FM2) (FM4) (FM8) (<FM64> (64text)) . . . (Appl. Text)].

For TTX message to TTX network management, the structure is:

[(FM0) <(FM9) (9text)> . . .].

Finally, for internal TTX Switch to Switch messages, the header structure is:

[(FM0) (Appl. Text)],

where the FM0 function code is 2x or Cx.

Continuing, the general rules of implementation for DIA messages in the preferred embodiment are as follows. All inter-messages are prefaced by a single FM0. Further, other header types can be optionally concatenated to the FM0. Also, headers should occur in ascending order by header type; i.e. FM0, FM2, FM4, FM8, FM9, FM64. Header and text length values are carried as binary values. Numeric fields contained within DIA headers are carried with the most significant values in the left-most byte(s).

5,347,632

26

Further, long gateway messages (greater than 1K bytes including headers) are sliced up into blocks. This segmentation is indicated by the presence of the FM2 Header. In the preferred embodiment, the current block number of the FM2 must be correctly set because it acts as a sequence number and provides a means to guarantee message integrity. In this regard, the total number of blocks field must be set correctly when sending the last block of a logical message. Receiving programs can determine end of message by testing block number=total number blocks. If the sender cannot pre-determine the total number of blocks in a beginning or middle of message block, the sender must place binary zeros in the total number of blocks field.

Still further, in the preferred embodiment, FM9 architected text may not span message blocks and may not be longer than 255 bytes. Additionally, FM64 architected text may not span message blocks and may not be longer than 512 bytes long. Yet further, only a single instance of FM2 and/or FM4 can be present in a message block. And, messages using FM9 or FM64 headers must be less than 1K bytes, and these messages should not be segmented into blocks.

Continuing with the DIA implementation rules, FM0 and FM2 must be present in each block of a multi-block message when being transported within the network system. Normal application message flow consists of a request/response pair. In normal processing, reception system applications send requests to host applications. Host applications return responses to these requests. The Reception System application initiates this dialogue. Sending nodes are responsible for inserting the proper "source id" (SID) and "destination id" (DID) into the FM0. Additionally, the communications manager (CM) of the reception system further described hereafter, acts on behalf of reception system transaction programs. Messages destined to the CM should be considered systems messages (FM0 FUNCTION=Cn). Messages destined to subordinate transactions on reception system 400 should be considered applications message (FM0 Function=On). Receiving nodes are responsible for swapping SID and DID contents when returning a response. Still further, intermediate nodes (with the exception of CICS switches and Gateways) need only be aware of FM0 and FM2 headers when routing messages to other destinations. CICS switches must be cognizant of all header layouts so that they can find the displacement to the transaction id which is contained within the first four bytes of application text. And server switch 205 provides a facility which allows responses to requests to be deliverable for at least a minimum period after the request was sent, e.g., one minute.

Finally, the preferred embodiment, CICS switches pass all DIA FM headers on to their subordinate applications. The applications are then responsible for returning the headers (with the SID/DID swap) back to the switch for responses. Both fixed length and variable length message headers are supported by the DIA. It must be noted that variable length headers are designed so that only the last field within the header is variable in length.

With regard to mode of conversation under utility sessions, the server switch 205 may engage in multiple sessions with an external CICS. Messages originating from network users may be routed through any of these sessions. Users are not forced to use the same utility session pipe for each message outbound to CICS. Pipes may be selected dynamically based on loading factors.

27

5,347,632

In a switch-driven environment CICS transactions may typically be initiated by means of start commands from the switch. In this arrangement, CICS transactions will pass outbound data back to the switch through a queue.

In accordance with DIA, the potentially dynamic nature of conversation routing dictates that CICS transaction programs not be written in a conversational mode. Rather, the transaction programs are preferably either pseudo-conversational or non-conversational. In this regard it should be noted that conversational transactions send a message and wait for a reply, and non-conversational transactions send a message and expect no reply. In the case of pseudo-conversational transactions, a message is sent, but no reply is expected. However, such messages are coded so as to be able to accept user input in various stages of completion, thus mimicking conversational transactions.

As will be appreciated by those skilled in the art, communications may arise within network 10 that do not require the standards applied to DIA messages. However, non-DIA messages are allowed in the DIA structure. Particularly, non-DIA messages are designated by setting the length portion of the header (i.e., the first byte) to binary zero. Considering header layout, and with input first to FMO headers, it should be noted that the FMO header provides routing information to both intermediate and boundary switches. In addition the FMO contains control fields which allow the sending application (which may be a switch) to communicate information to the switch which "owns" the destination application. When an originating application wishes to converse with an application resident on the other side of an utility session it must initially pass an FMO header with a function code representing an "begin session" to its controlling switch. The begin session code requests the assistance of any intervening switches in the establishment of an application session between the requestor and the destination application specified in the DID.

When either application session partner wishes to terminate its conversation the session partner must pass an FMO header to its switch, specifying either a function code representing an "end session", or "end session abnormal", or "request terminate". These function codes request the assistance of any intervening switches in the termination of the application session between the requestor and the destination application specified in the DID. In this arrangement an end session function code is unconditional and does not require an acknowledgment. An end session abnormal function code is unconditional and does not require an acknowledgment. And, a request terminate function code is conditional and requires a positive acknowledgement. The positive acknowledgement to a request terminate is an end session. The negative acknowledgement to a request terminate is a function code representing "status Message".

Further, "status/return" function codes "system up", "system down", "echo", "system message" are used by corresponding applications in different regions of network 10 to determine application availability and user session status. Function codes are also used to designate end-to-end user message classes of service. These classes of service refer to a delivery requirement classification and are distinguished from SNA COS. Network class of service allows applications to specify whether or not responses to requests can be delivered after the standard timeout of server 205 has occurred.

28

In accordance with the invention, the DIA headers are arranged in a predetermined form base on their function. More particularly, FMO headers, also known as Type "O" headers are required for every message within the network. Header Type O provides information necessary for routing and message correlation. Its fields include:

Header Length	Length of header data including length field.
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.
Function Code	Contains message function.
Data Mode	Indicates attributes of message data. The "response expected" bit should be turned off if no response is expected, for instance, when sending the response to a request.
Source Id	Identification of end-user sending current message
Logon Sequence Number	number which in conjunction with source id provides unique identification of source when source is reception system 400.
Message Sequence Number	used to correlate requests and responses.
Destination Id	Identification of message destination. All messages are routed by destination id. When responses to messages are sent back to original source, the source id and destination id fields must be swapped.
Text Length	length of all remaining data in the message to the right of this fields. (Includes length of concatenated headers if any are present).

The layout for the Type O header is as follows:

Header Type 0 layout:	
Byte 0	Header Length (hexadecimal)
Byte 1	Header Type
bit 0	no other headers present; or concatenated header present
bits 1-7	current header type
Byte 2	Function Code; i.e.
	Application message (Class of Service)
	Status/Return Code message
	Begin Session
	End Session (normal)
	End Session (error)
	Clear Request (request terminate)
	System Up
	System Down
	Echo
	System Message
	Prepare to bring System Down
Byte 3	Data Mode (bit flags)
bits 0-7	Compaction;
	Encryption;
	Response Expected;
	Response;
	Unsolicited Message;
	Logging required;
	Timeout Message Required;
	Reserved;
Bytes 4-7	Source ID
bits 0-7	Region ID (hexadecimal)
bits 8-19	xxxx xxxx xxxx
	Unit: Source application id if in Application mode
	xxxx xxxx xxxx
	Unit: Source Concentrator unit if in Reception System mode
	xxxx Id Mode e.g.,
	Reception mode
	Reception mode
	Server 205 Application mode
	Server 205 Application mode
	Cache 302 Application mode
	Reserved
bits 20-23	xxxx xxxx Sub-unit ID (hexadecimal)
bits 24-31	Logon Sequence Number (hexadecimal)
Byte 8	Message Sequence Number (hexadecimal)
Byte 9	Destination ID
Bytes 10-13	Region ID (hexadecimal)
bits 0-7	xxxx xxxx xxxx
bits 8-19	Unit: Destination application ID

-continued

	if in Application mode
	xxxx xxxx xxxx
	Unit: Destination Concentrator
	if in Reception System mode
bits 20-23	xxxx Id Mode; e.g., Reception mode Reception mode Server 205 Application mode Server 205 Application mode Cache 302 Application mode Reserved
bits 24-31	xxxx xxxx
	Sub-unit ID (hexadecimal)
Bytes 14-15	Text Length.

With regard to FM2 or Type 2 messages, when an application is transmitting a large message, the sending application or its controlling switch can slice up the message into a number of smaller messages. The FM2 message header is used to indicate how these smaller messages can be reassembled into a single logical message by the receiving application or its controlling switch.

In preferred form, the maximum logical message size is 64K. The maximum message block size is 1K including all headers. Block sequence numbers in the FM2 range from 1 to a maximum of 255. And a single block message will be sequenced as block 1 of 1 in the FM2.

When network objects are large (greater than 1K bytes) they are sliced up into smaller blocks. Each object block is prefaced by an "object block header". Object block headers are found in the application text portion of a message. Object block headers provide sequencing information to cache/concentrator 302. The presence of an object block header does not obviate the requirement for an FM2 DIA header, except in the case of messages from the cache/concentrator down to RS 400. Both an object block header and a FM2 may be present in a message. Sequence numbering within object block headers ranges from 0 to 255. A single block Object will be sequenced as block 0 of 0.

Messages larger than 1K are subdivided into 1K blocks when being transmitted between the server switch 205, cache/concentrators 302, and reception systems 400.

Header Type 2 (FM2) message header contain information about this dividing of large messages and is useful when re-constructing large messages. The fields for an FM2 message header are as follows:

Header Length	length of header data including length field.
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.
Number of Blocks	total number of blocks used to transmit the logical message. If the total number of blocks cannot be determined at the time the first or middle blocks of a message are being sent, this field may be set to zero. The last block of a message must contain the correct total number of blocks.
Block Number	number of the current message block being transmitted.

The layout for a Type 2 header is as follows:

Byte 0	Header Length (hexadecimal)
Byte 1	Header Type
bit 0	no other headers present; or concatenated header present

-continued

bits 1-7	current header type
Byte 2	Number of Blocks (hexadecimal)
Byte 3	Current Block Number (hexadecimal).

With regard to FM4 type headers, also referred to as Type "4", these headers have been designed for communications between network gateway interface applications and external computer systems. For Type 4 Headers, the fields are as follows:

Header Length	length of header data including length field.
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.
Network User	a seven byte field containing the internal ID of the network user on whose behalf a conversation is being held with the external computer system.
External Data Correlation Id	Reserved Mode a field reserved for use by the external computer system. The contents of this field will initially be set to zero when a conversation is initiated across a gateway. The external system may then set the contents of this field to any value desired. Subsequent messages originating from TTX within the bounds of a virtual subscriber to external host session will echo the contents of the Correlation Id field back to the external system.

The layout for a Type 4 header is as follows:

Byte 0	Header Length (hexadecimal)
Byte 1	Header Type
bit 0	no other headers present; or concatenated header present
bits 1-7	current header type
Bytes 2-8	Network User Id (ASCII)
Byte 9	External Data Mode
	0000 0000 Reserved
Bytes 10-n	Correlation Id (binary, max length=8 bytes).

Next are FM8 or Type 8 headers. Type 8 headers have been designed to provide secondary routing destinations. Their fields are as follows:

Header Length	length of header data including length field.
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.
Secondary Destination	a symbolic name representing the ultimate destination for the message.
The layout for Type 8 header is:	
Byte 0	Header Length (hexadecimal)
Byte 1	Header Type
bit 0	no other headers present; or concatenated header present
bits 1-7	current header type
Bytes 2-9	Symbolic Destination Name

For FM9 or Type 9 headers, the header has been designed to communicate to a VTAM application which provides various network management support functions. More specifically, the VTAM application has been developed in order to provide a general network management interface which both supports the network (by means of the DIA) and simplifies its maintenance. Additionally, VTAM application provides data transfer and remote functions, the ability to write to, and read from, a centrally located and maintained database in order to archive statistics and other inter-network mes-

sages, and formatting of binary data into Hexadecimal Display.

Header Length	length of header data including length field.	5
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.	
Function Code	indicates general message type.	10
Reason Code	indicates message content.	
Flags	indicates application action to be performed.	
Text Length	indicates length of subsequent text message. (Not including possible concatenated headers)	

The layout for type 9 headers is:

Byte 0	Header Length (hexadecimal)	5
Byte 1	Header Type	
bit 0	no other headers present; or concatenated header present	10
bits 1-7	current header type	
Byte 2	Function Code; e.g. Command Statistics Alert Control	10
Byte 3	Reason Code Backbone Alerts Message Reception-originated Alerts Message	
Byte 4	Flags	10
bits 0-3	Store by Key - 8 char. name follows; Retrieve by Key - 8 char. name follows; Data is Binary; Data is ASCII; Data is EBCDIC Reserved	
bits 4-7	Reserved	10
Byte 5	Text Length if Flags = 1 . . . or . 1 . . . then chars 0-7 should be the storage key. It is recommended that record storage keys initially be the same as the Resource Name to which the data pertains.)	

In the case of FM64 or Type 64 headers, the headers are used to transmit error and status messages between applications. Intermediate nodes need not examine the contents of the FM64 headers except in the case of the CICS switch which must obtain the displacement to the application text. If applications subordinate to an application subsystem are not available, the subsystem would strip the application text from the message, concatenate an FM64 message to any other headers which are present in the inbound message, and return the message to its original source.

Header Type 64 has been designed for the communication of status information between users, and prefaces architected message text. The fields for Type 9 headers are:

Header Length	length of header data including length field.	55
Header Type	Bit 0 is header concatenation flag. Bits 1-7 indicate current header type.	
Status Type	indicates rype or status communicated such as status request or error.	60
Data Mode	indicates whether message text is ASCII or EBCDIC	
Text Length	Length of subsequent message text (Not including possible concatenated headers).	

The header Type 64 layout is:

Byte 0	Header Length (hexadecimal)
Byte 1	Header Type

-continued

bit 0	no other headers present; or concatenated header present	10
bits 1-7	current header type	
Byte 2	Status Type Information Status Request Error Terminate	10
Byte 3	Data Mode; e.g., EBCDIC ASCII Binary	
Bytes 4-5	Text Length	

In accordance with the invention, it has been determined that in some cases it is desirable to pre-define certain application level message formats so that they may be consistently used and interpreted. The following discussion is devoted to architected message text formats which are processed at the application level. For FM9 message text, in order to accommodate "Reliability Serviceability Availability" (RSA) functions within network 10, a fixed format for "alerts" is defined in the preferred embodiment. Particularly if it is defined as message text following an FM9 header. The FM9 Function Code Alerts Message would be as follows:

Byte 0	Reserved value	30
Byte 1	System Origin	
Byte 2	Internal/External flag	30
Byte 3-5	Message Originator	
Byte 6-9	Message Number	35
Byte 10	Severity indicator; e.g. Error Information Severe Error Recovery Successful Warning	
Byte 11	Reserved value	35
Byte 12-14	Error Threshold.	

For FM64 message text, the application message text is always prefaced by the appropriate header which indicates whether message text is ASCII or EBCDIC. The FM64 message text fields are as follows:

Action Field	indicates type of operator or application action to b performed	55
Module Name	Sending application Id Format of this field is SSSSTnnnn where SSS = sender initials T = type 0 = Network standard for all gateways 1 = non-standard, gateway specific nnnn = Sender Site number	
Reference Number	Number assigned by sender for reference This number is used to indicate specific error codes if the message is an error message (FM64 stat type 8). This number is used to indicate specific commands if the message is a status request (FM64 stat type 4).	60
Text	Alphanumeric (Printable) text.	

The FM64 Message Text layout is:

Byte 0	Action Field (alphanumeric), e.g., Action Decision Information Wait	65
Bytes 1-8	Module Name (alphanumeric)	

33

-continued

Bytes 9-12	Reference Number (display numeric)
	Default
	request user status
	user active
	user inactive
	user inactive - retry after interval
	store in user mailbox
	cache to server link failure
	request appl status
	server to host failure
	appl active
	appl inactive
	appl inactive - retry after interval
	message was undeliverable
	response was timed out
	syncpoint
	checkpoint
	delay
	appl. error codes
Bytes 13-n	Text (alphanumeric).

Turning next to co called "Backbone States", as will be described below, application sessions may be used as pipes for user transaction traffic. In this regard, it is desirable to establish a set of protocols to be used between originating users and destination users. Further it is important for intermediate nodes to be aware of the status of connectivity with adjacent nodes and specifies some actions to take when messages are known to be undeliverable.

In this context, it is to be noted that the "system up" message is used to signal the start of application traffic between the switch applications. The originating application transmits an FMO with a system up function code and response expected. The receiving application swaps the SID/DID, sets the Response bit on, and returns the message. If the receiving application is not available no response will be returned and the message will time out.

In the case of "system down" messages, the message is used to prepare the termination of the session between switch applications. The originating application transmits an FMO with a session down function code and response expected. The originating application sends an FM64 with "status type=terminate", and data mode=EBCDIC. FM64 text follows the header with "action field"=A (Action), "module name"=SSSx0nnnn, "reference number"=0, Text=((timestamp=HHMMSS), Number of current users=NNNNN). The intended result is that the originating application will not accept any messages inbound to the utility session. The responding application will then have the opportunity to return outstanding responses across the utility session. The responding application then returns an FMO with System Down back to the originating application.

For each "echo" messages, the echo message may be used to determine whether a major application is still available. Specifically, the originating application sends an application message to its gatewayed partner using a FMO with an echo function. The destination application swaps the SID/DID, set the response bit on and returns the message otherwise untouched, thus effecting echo. For "APPL status request messages, the message is used to determine the status of a major application between nodes.

Continuing, for "unsolicited application status posting" messages these messages are used for transmission of application status messages by unsolicited application (No response expected) across a nodes. For the message, the originating application wishes to post an application status to its partner in another node. This mes-

5,347,632

34

sage may be on the behalf of the originating application itself or on behalf of another application.

Turning next to user to internal APPL messages, and with regard to "session beginning" it is to be noted these messages normally arise at the start of conversation between a user and an internal application. For them the network user sends an FMO with a "begin session" function code and "response expected". The responding application swaps the SID/DID, supplies a "correlation Id", and returns both the FMO with the response bit set.

In the case of rejection of a conversation initiation requests, the originating application transmits an FMO with a "begin session" function code and "response expected". The responding application swaps the SID/DID, and returns the FMO with the response bit set as well as a function code of "abend" session.

For "applications" messages, these messages normally arise at the middle of conversation between a network user and an internal application. In this case, the originating user transmits an FMO with an "application" message function code, and "response expected". The responding application swaps the SID/DID, sets the response bit on and returns the response.

"End session" messages typically arise in connection with unconditional termination of user/internal application sessions. The originating transmits an FMO with an "end sessions" function code. Here however, no response is expected from the corresponding application.

For an "end session abnormal" message, the message unconditionally terminates an application conversation "abend".

Continuing, "request terminate" messages cause conditional termination of session with an internal application. For messages concerning "rejection of a request due to link failure", in the case of server 205 to host link, the originating application transmits and FMO with "response expected". The message is intercepted by server 205 which recognizes it as undeliverable. A server 205 application returns the message with an FM64 message after stripping the application text.

For messages concerning rejection of request due to link failure, in the case of communication between the cache/concentrator 302 and server 205, the originating application transmits an FMO with Response Expected. The message is intercepted by the cache/concentrator 302 which recognizes it as undeliverable. A cache/concentrator application returns the message with an FM64 message after stripping the application text.

For messages concerning "conditional terminate rejected", the message is issued where a conditional termination of application conversation is not accepted by partner application.

For "user continuity posting" messages, the message is used where the originating application wishes to post the status of a user to its partner application across the gateway 210.

Continuing, for "user continuity requests", the message is used where an external application requests logon status of a particular network user.

In the case of "application error" messages, the message is used where transmission of application error message by responding application is required.

Still further, for "timeout scenarios", and specifically, "timeout scenario with timeout response required", the originating user sends an application message to an internal application with "data mode". . . "response

5,347,632

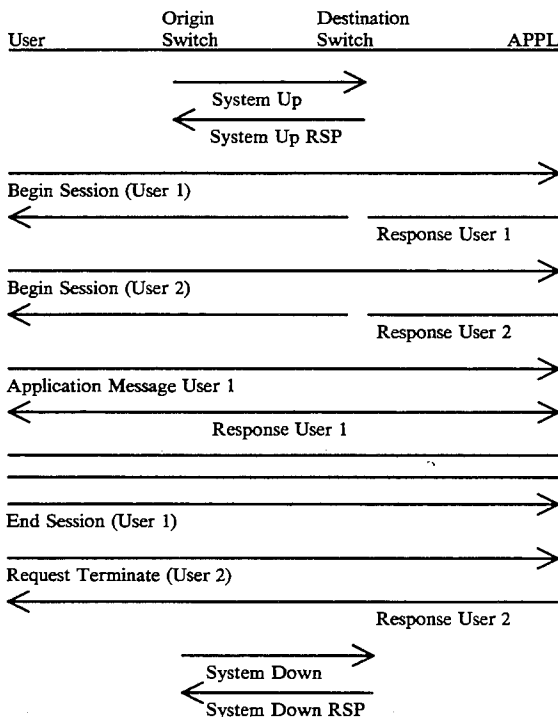
35

expected" and "timeout response" required. The originating switch sets a timer for each "response expected" outbound message. If a response is not received before the switch timeout value is reached the switch 205 sends a message with an FM64 header having a "timeout reference" code to the originating application.

For "response occurs after timeout" messages, the originating user sends an application message to an internal application with "response expected". The originating switch sets a timer for each "response expected" outbound message. If a response is received after the timeout value is exceeded, server 205 switch routes the message to a server 205 application which may log the message as non-deliverable, ship the message to the user, or drop it depending on the FMO class of service option specified on the original request message.

In the case of "maximum resources scenario" messages, the originating user transmits a message to a destined internal application. The destination switch determines that no resources are currently available to support the transmission, and returns the message to the originator, after inserting an FM64 with a "status=error" and FM64 text with an "action" wait. The originating user may then retry or take other action.

Finally, the following graphic example illustrates normal message flow.



Turning next to messages passed over gateways 210, the normal exchange of messages between the network and external parties occurs between two applications; i.e., the server 205 network message handler (NMH). The server Switch 205 is an application which is written and maintained by network 10 and resides on it. The message handler resides on the other side of gateway 210 from network 10 and may be written and maintained by the external party; i.e., suppliers of information to network 10 such as Dow Jones.

The session between the two applications is used as a pipe for the communications between many network

36

users and a variety of applications external to the network. In this design, the switch server 205 has three primary responsibilities. It must pass network originated messages across the gateway to the network message handler. It must distribute messages returning across gateway 210 to the appropriate network applications or users, i.e. RS 400. Additionally, it must manage the continuity of a network user session with the external service provider. Typically, users enter into a conversation with a set of transactions. This set of transactions is referred to as a task. These tasks are called user sessions. The boundaries of these tasks are indicated by begin session/end session flags.

The network message handler also has several responsibilities. It must pass externally originated messages across gateway 210 to the switch server 205 at network 10. It must distributed messages returning across gateway 210 to the appropriate external applications. And, it must be able to communicate the availability of external applications to network switch server 205.

With regard to gateway messages, in the case of "application to application" messages, and for "system up" messages, the system up message is used to signal the start of application traffic between switch 205 and the network message handler. The originating application transmits an FMO with function code "system up", and "response expected". The receiving application swaps the SID/DID, sets the response bit on, and returns the message. If the receiving application is not available no response will be returned and the message will time out.

Continuing for gateway "system down" messages, the system down message is used to prepare the termination of the session between the switch 205 and the NMH. The originating application transmits an FMO with function code "session down" and "response expected". The originating application sends an FM64 with "status type"="terminate" "data mode"="EBCDIC". FM64 Text follows the header with "action field"="A" (Action), "module name"="SSSx0nnnn", "reference number"="0", "text"=((timestamp=HHMMSS), number of current users=NNNNN). The intended result is that the originating application will not accept any messages inbound to the utility session. The responding application will then have the opportunity to return outstanding responses across the utility session. The responding application then returns an FMO with system down back to the originating application.

Further, for "prepare to bring system down" messages, the message is used to prepare the termination of the session between the Switch 205 and the NMH. The originating application transmits an FMO with function code "prepare system down". The responding application transmits an FMO with function code "session down" and "response expected". The responding application sends an FM64 with "status type"="terminate", "data mode"="EBCDIC" FM64 Text follows the header with "action field"="A" (action), "module Name"="SSSx0nnnn", "reference number"="0", "text"=((Timestamp=HHMMSS), number of current users=NNNNN). The intended result is that the responding application will not accept any messages inbound to the utility session. The originating application will then have the opportunity to return outstanding responses across the utility session. The originating

37

5,347,632

application then returns an FM0 with "system down" back to the responding application.

For "echo" messages, the message may be used to determine whether a major application is still available. The originating application sends an application message to its gatewayed partner using a FM0 with function echo. The destination application swaps the SID/DID, set the response bit on and returns the message otherwise untouched.

In the case of "APPL status request", the request is used to determine the status of a major application across the gateway.

Continuing, for "unsolicited application status posting messages, the message is used for transmission of application status messages by unsolicited applications no response expected across a gateway. In this case the originating application wishes to post an application status to its partner across the gateway. This message may be on the behalf of the originating application itself or on behalf of another application. For network to use "external APPL" messages, within the case of "begin session" messages, the message is used for normal start of conversation between a and an external application. The user, i.e. RS 400 sends an FM0 with function "begin session" and "response expected", as well as an FM4 with null value in the "correlation id". The responding application swaps the SID/DID, supplies a Correlation ID, and returns both the FM0 with the response bit set and the FM4.

For rejection of a conversation initiation request, the originating application resident application, transmits an FM0 with function Begin Session and Response Expected as well as an FM4 with NULL value in the Correlation ID. The responding application swaps the SID/DID, and returns the FM0 with the response bit set as well as a function code of ABEND session. The responding application also returns the FM4.

Further, for "applications" message, the message is used for normal middle of conversation between a network user and an external application. The originating user transmits an FM0 with function code "application" message, and "response expected". It also supplies the TTXUID and the correlation id received on the begin session response back to the corresponding application across the gateway. The responding application swaps the SID/DID, sets the response bit on and returns the FM0 and FM4.

For "end session" message, the message is used for unconditional termination of user/external application sessions. The originating user transmits an FM0 with function code "end session", no "response expected". Additionally it sends an FM4 containing the TTXUID and the echoed "correlation id" in an FM4. No response is expected from the corresponding application.

For "end session abnormal" messages, the message is used for unconditional termination ABEND of gatewayed application conversation.

In the case of "request terminate", the message is used for conditional termination of user session with an external application.

For "conditional terminate rejected" messages, the message is used for a conditional termination of application conversation not accepted by partner application across a gateway.

For "user continuity posting" messages, the message is used where the originating application wishes to post the status of a user to its partner application across the gateway.

38

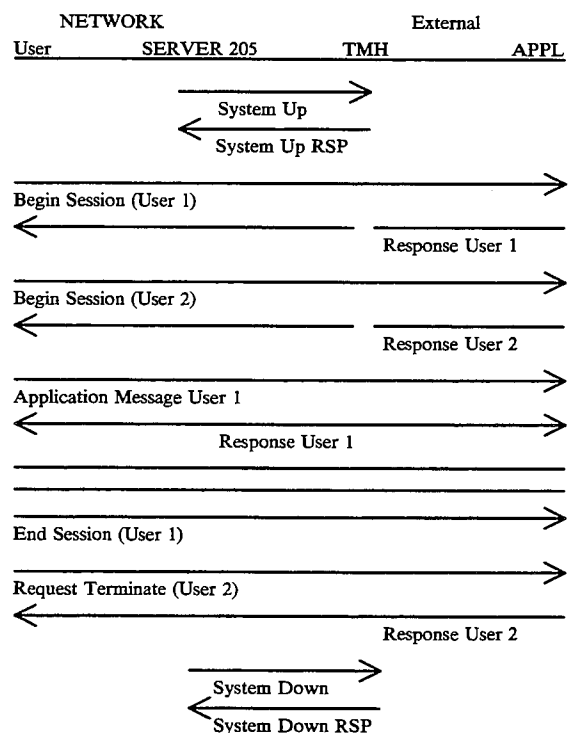
In the case of "user continuity" request, external application requests logon status of a particular user, i.e. RS 400. For "application error" messages, the message is used for transmission of application an error message by responding application across a gateway.

In the case of "delayed response" messages, the originating application sends an application message to its gatewayed partner using the minimally a FM0 and a FM4 FM64 may be present. The destination switch signals an application on the originating side that the response may be slow by sending a FM0 with function code "status/return", the response bit is not set. The FM4 is returned, and an FM64 "status", FM64 text "Action"="Information" is also sent. Slow response may be due to a number of factors such as function shipping requirements or many I/Os. In parallel, the gateway partner application processes the message according to normal flow.

For "timeout scenario", the originating user sends an application message to an external application with "response expected". The switch server sets a timer for each "response expected" outbound message. If a response is received after the timeout value is exceeded, the TPF switch routes the message to a TPF application which may log the message as non-deliverable, ship the message to the user, or drop it depending on the FM0 class of service option specified on the original request message.

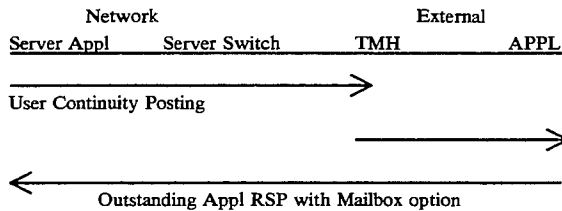
For the "maximum resources scenario" messages, the originating user transmits a message to a destined external application. The network message handler determines that no resources are currently available to support this transmission. The network message handler returns the message to the originator, after inserting an FM64 with a "Status"="Error" and FM64 text with an "action=wait". The originating user may then retry or take other action.

Finally, an example illustrates normal message flow.



39

And, the following is an example that illustrates premature loss of user connectivity due to the loss of connection between the network switch sever 205 and a cache/concentrator 302. In this case, an application peripheral to switch 205 posts the user status inactive to the NMH using an FM64 Ref=008 user inactive. External application reaction to this posting is implementation dependent. In this example, the external application returns outstanding responses using the FM64 "ref"='mailbox option'.



OBJECT LANGUAGE

In accordance with the invention, in order to enable the manipulation of the network objects, the application programs necessary to support the interactive text/5 graphic sessions are written in a high-level language referred to as "TBOL", (TRINTEX Basic Object Language, "TRINTEX" being the former company name of one of the assignees of this invention). TBOL is specifically adapted for writing the application programs so that the programs may be compiled into a compact data stream that can be interpreted by the application software operating in the user personal computer, the application software being designed to establish the network Reception System 400 previously noted and described in more detail hereafter.

In accordance with the invention, the Reception System application software supports an interactive text/graphics sessions by managing objects. As explained above, objects specify the format and provide the content; i.e., the text and graphics, displayed on the user's screen so as to make up the pages that constitute the application. As also explained, pages are divided into separate areas called "partitions" by certain objects, while certain other objects describe windows which can be opened on the pages. Further, still other objects contain TBOL application programs which facilitate the data processing necessary to present the pages and their associated text and graphics.

As noted, the object architecture allows logical events to be specified in the object definitions. An example of a logical event is the completion of data entry on a screen; i.e., an application page. Logical events are mapped to physical events such as the user pressing the <ENTER> key on the keyboard. Other logical events might be the initial display of a screen page or the completion of data entry in a field. Logical events specified in page and window object definitions can be associated with the call of TBOL program objects.

Reception Systems 400 is aware of the occurrence of all physical events during the interactive text/graphic sessions. When a physical event such as depression of the forward tab key corresponds to a logical event such as completion of data entry in a field, the appropriate TBOL program is executed if specified in the object definition. Accordingly, the TBOL programs can be thought of as routines which are given control to perform initialization and post-processing application logic

5,347,632

40

associated with the fields, partitions and screens at the text/graphic sessions.

Reception System 400 run time environment uses the TBOL programs and their high-level commands called verbs to provide all the system services needed to support a text/graphic session, particularly, display management, user input, local and remote data access.

In accordance with the invention, the TBOL programs have a structure that includes three sections: a header section in which the program name is specified; a data section in which the data structure the program will use are defined; and a code section in which the program logic is provided composed of one or more procedures. More specifically, the code section procedures are composed of procedure statements, each of which begins with a TBOL key word called a verb.

In accordance with the invention, the name of a procedure can also be used as the verb in a procedure statement exactly as if it were a TBOL key word verb. This feature enables a programmer to extend the language vocabulary to include a customized application-oriented verb commands.

Continuing, TBOL programs have a program syntax that includes a series of "identifiers" which are the names and labels assigned to programs, procedures, and data structures.

An identifier may be up to 31 characters long; contain only uppercase or lowercase letters A through Z, digits 0 through 9, and/or the special character underscore (_); and must begin with a letter. Included among the system identifiers are: "header section identifiers" used in the header section for the program name; "data section identifiers" used in the data section for data structure names, field names and array names; and finally, "code section identifiers" used in the code section for identification of procedure names and statement labels.

The TBOL statement syntax adheres to the following conventions. Words in uppercase letters are key words and must be entered exactly as shown in an actual statement. When operand are allowed, descriptive operand names and lowercase letters follow the key word. In this arrangement, operand names or laterals are entered in an actual statement. Operand names enclosed in square brackets ([]) are optional and are not required in an actual statement. Operand names separated by a bar (|) mean that one, and only one, of the separated operand can be included in an actual statement. Operand names followed by an ellipsis (...) can be entered 1 or more times in an actual statement. Model statement words not separated by punctuation must be separated by at least one blank (or space character) in actual statements. Model statement punctuation such as comma (,) , semicolon (;) , less than sign (<) , equal sign (=) , greater- than (>) , and parentheses (()) must be included where shown in actual statements. Square brackets ([]) , bars (|) , and ellipses (...) should not be included in actual statements.

An example of a model statement would be as follows:

GOTO_DEPENDING_ON index, label (.label . . .) .

This model says that a valid GOTO_DEPENDING_ON statement must begin with the word "GOTO_DEPENDING_ON" followed by at least one blank. Thereafter, an "index" and a "label" separated by a comma must be included. The index and at least one label are required. Additional labels may also

be used, provided each is preceded by a comma. Further, the statement must have a semicolon as the last character.

Comments can be included in a TBOL program on a statement line after the terminating semicolon character or on a separate comment line. Comment text is enclosed in braces ({}). For example: {comments are enclosed in braces}. Comments can be placed anywhere in the source code stream since, in accordance with the invention they are ignored by the TBOL compiler. Additionally, blanks (or space characters) are ignored in TBOL statement lines except where they function as field separators.

As noted, TBOL programs have a structure that includes a header section, data section and code section. More particularly, every TBOL program must have a header section. The header section contains a PROGRAM statement. The PROGRAM statement contains the key word PROGRAM followed by the name of the program. For example:

PROGRAM program_name; where "program_name" is an identifier; i.e., the name of the program.

Accordingly, the header section for a TBOL program called LOGON would look like as follows:

PROGRAM LOGON: {User logon program}

The data section in a TBOL program begins with the key word DATA which is followed by data structure statements. The structure statements contain the data structure definitions used by the program. If the data structure does not have to be defined for the program it can be omitted. However, if a TBOL program does not include a data section, it must use a more restricted structure, more fully explained hereafter. As an example, the data syntax would be:

DATA structure [structure . . .];

where "structure" is a data structure statement. The data structure statement contains a definition, which consists of the data structure name followed by an equal sign and then the names of one or more variables. For example:

structure_name=variable_name [,variable_name . . .];

where "structure_name" is an identifier; i.e., the name of the data structure; and "variable_name" is an identifier for the variable; i.e., the name of a variable.

All of the variables in the data structures are defined as string (or character) variables. TBOL string variables are of two kinds, fields and arrays. In the case of filed definitions, a variable field is defined with an identifier; i.e., the name of the field. No data type or length specification is required. An individual field is referenced by using the field name. Further, subsequent fields can be referenced by using a field name followed by a numeric subscript enclosed in parentheses (()). The subscript however, must be an integer number.

A field name followed by a subscript refers to a following field in the data section of a TBOL program.

The subscript base is 1. For example, if a field CUST_NBR were defined, then CUST_NBR refers to the field CUST_NBR, CUST_NBR(1) also refers to the field CUST_NBR and CUST_NBR(2) refers to the field following CUST_NBR.

In the case of array definitions, the TBOL array is a one-dimensional table (or list) of variable fields, which can be referenced with a single name. Each field in the array is called an element.

An array can be defined with an identifier, particularly, the name of the array, followed by the array's dimension enclosed in parentheses (()). The dimension specifies the number of elements in the array. By way of illustration, if an array is defined with a dimension of 12, it will have 12 elements. An individual element in an array is referenced by using the array name followed by a numeric subscript enclosed in parentheses (()). The subscript indicates the position of the element in the array. The first element in an array is referenced with a subscript of 1. The subscript can be specified as either an integer number or an integer register as described, hereafter.

With regards to variable data, data contained in variables is always left-adjusted. Arithmetic operations can be formed on character strings in variables if they are numbers. A number is a character string that may contain only numeric characters 0 through 9, an optional decimal point, an optional minus sign in the left-most position, commas and the dollar sign (\$).

When you perform an arithmetic operation on a character string, leading and trailing zeros are trimmed and fractions are truncated after 13 decimal places. Integer results do not contain a decimal point. Negative results contain a minus sign (-) in the left-most position.

Each field and each array element has a length attribute which is initialized to zero by the Reception System at program start-up. The LENGTH verb, to be described more fully hereafter, can be used to set the current length of a field or array element during program execution. The maximum length of a field or an array element is 65,535.

Further, the maximum number of variables that can be defined in the data section of a TBOL program is 222. This number includes fields and array elements.

The following example data section contains five data structure statements, each defining a data structure. Each structure statement begins with the name of the data structure followed by an equal sign.

Next, are the names of the variables which make up the structure. The variable names are separated by commas. The last variable name in each structure statement is followed by a semicolon which terminates the statement.

The third data structure given, i.e. SALES_TABLE, contains two arrays. The others contain fields. The last structure statement, i.e. WK_AREA is an example of a single line.

DATA	{Key word DATA begins data section}
BILL_ADDR=	{data structure BILL_ADDR}
BILL_NAME,	{field1 BILL_NAME}
BILL_ADDR1,	{field2 BILL_ADDR1}
BILL_ADDR2,	{field3 BILL_ADDR2}
BILL_ADDR3,	{field 4 BILL_ADDR3}
SHIP_ADDR=	{data structure SHIP_ADDR}
SHIP_NAME,	{field1 SHIP_NAME}
SHIP_ADDR1,	{field2 SHIP_ADDR1}
SHIP_ADDR2,	{field3 SHIP_ADDR1}

-continued

SHIP_ADDR3,	{field4 SHIP_ADDR1}
SALES TABLE=	{data structure SALES_TABLE}
MONTH QUOTA(12),	{array1 MONTH_QUOTA}
MONTH SALES(12),	{array2 MONTH_SALES}
MISC_DATA=	{data structure MISC_DATA}
SALESPERS_NAME,	{field1 SALESPERS_NAME}
CUST_TELNBR;	{field2 CUST_TELNBR}
WK_AREA=	{data structure WK_AREA}
TEMP1,	
TEMP1;	

Continuing, TBOL contains a number of predefined data structures which can be used in a TBOL program even though they are not defined in the program's data section. There are two kinds of TBOL-defined data structures, these are "system registers" and "external data structures".

In the case of systems registers, three different types exist. The first type are termed "integer registers", and are used primarily for integer arithmetic. However, these registers are also useful for field or array subscripts. The second type are termed "decimal registers" and are used for decimal arithmetic. The third type are called, "parameter registers" and are used to pass the data contained in procedure statement operand when the name of a procedure is used as the verb in the statement rather than a TBOL keyword.

The variables defined in the data section of a program are string (or character) variables, and the data in them is kept in string format. In most cases there is no need to convert this data to another format, since TBOL allows substantially any kind of operation (including arithmetic) on the data in string form. As will be appreciated by those skilled in the art, this eliminates the clerical chore of keeping track of data types and data conversion.

There are some cases where it is desirable to maintain numeric data in binary integer or internal decimal format. For example, an application involving a great deal of computation will execute more efficiently if the arithmetic is done in binary integer or internal decimal format data rather than string data. In these cases, data conversion can be performed by simply moving the numeric data to the appropriate register. When data is moved from a register to a variable, it is converted to string format.

Integer registers are special-purpose fields for storing and operating on integer numeric data in binary format. The integer registers are named I1 through I8. Numeric data moved to an integer register is converted to an integer number in binary format. Further, an attempt to move non-numeric data to an integer register will cause an error. The largest negative number an integer register can hold is -32,767, while the largest positive number than can be held is 32,767. An noted arithmetic operations in integer registers will execute more efficiently than arithmetic operations in string variables.

Decimal registers are special-purpose fields for storing and operating on numeric data in internal decimal format. The decimal registers are named D1 through D8. Numeric data moved to a decimal register is converted to a decimal number in internal decimal format. An attempt to move non-numeric data to a decimal register will cause an error. The largest negative number a decimal register can hold is -9999999999.9999999999, while the largest positive number a decimal register can hold is 9999999999.9999999999. Additionally, decimal registers can not be used as field or array subscripts.

And, again, arithmetic operations in decimal registers will perform better than arithmetic operations in string variables.

As pointed out above, the code section of a TBOL program contains the program logic, which itself is composed of one or more procedures. In the logic, the procedures are expressed as procedure statements. Each procedure statement begins with a TBOL keyword called a verb which is followed by operand, or parameters containing the data on which the verb is to operate. The name of a procedure can be used as the verb in a procedure statement exactly as if it were a TBOL keyword verb. As noted this enables the creator of a TBOL program; i.e. the party creating the text/graphic session, to extend the language vocabulary to include his own application-oriented verb commands.

When a procedure is used as the verb in a procedure statement, TBOL saves the current parameter register values, and the parameter data in the verb operand is moved into the parameter registers where it is available to the "called" procedure. When the "called" procedure returns, TBOL restores the saved parameter register values.

Parameter registers are special-purpose fields for passing parameter data to "called" procedures. The parameter registers are named P0 through P8. When a procedure is "called" by using its name as the verb in a procedure statement, the current contents of P0 through P8 are saved. Further, data from the first operand in the procedure statement is placed in P1; data from the second operand is placed in P2; and so on, up to eight operand. If no operand, or less than eight operands are specified, the parameter registers corresponding to the missing operand are set to null. In accordance with this arrangement, the number of operand is placed in P0, and the "called" procedure is given control.

When control returns to the "calling" procedure from the "called" procedure, the previous contents of P0 through P8 are restored. Following execution of the "called" procedure, execution of the "callings" procedure continues.

The "calling" procedure can pass along its own parameters to the "called" procedure by naming parameter registers as operand. The TBOL internal stack can be used to pass additional data to the "called" procedure, or to pass data back to the "calling" procedure.

There are two kind of TBOL-defined external data structures; they are partition structures and global structures. With regard to partition external data structures, as noted above the screens displayed during a test/graphic session are called pages. As also noted, pages may be divided into separate areas called "partitions". Each page partition has its own predefined partition external data structure. Each partition external data structure can contain up to 256 variables for data pertaining to that partition. A TBOL program associated

5,347,632

45

with a particular partition has access to the partition's external data structure and the variables it contains. However, the program cannot access another partition's external data structure.

The variable in a partition external data structure are character string variables like those defined in the data section of a program. The variables within each partition external data structure are named &1 through &256. The DEFINE compiler directive enables the program to use meaningful names for these-variables in the program source code.

Partition external variables are used to hold screen field data, program flow data and applications data. In the case of screen field data, when page and window objects are defined, the fields in the screen partitions are assigned to partition external variables. The TBOL Object Linker resolves these references and at program execution time the Reception System transfers data between the screen fields and their associated partition external variables. The TBOL program has access to the variables, which contain the data entered in the screen fields by the user, and the user has access to the screen fields of which contain the data placed in the variables by the program.

For program flow data, partition external variables are used to hold the object identifiers needed by a TBOL program for transferring control. These may be page object identifiers for transfer to another text/graphic screen page, or window object identifiers needed to open a window on the current page. As in the case of screen field data, flow data values are placed in partition external variable by the TBOL Object Linker.

Finally, for application data, partition external variables can be used to hold partition-specific application data such as tables of information needed by the program to process the expected screen field input.

With regard to the global external data structure, the predefined global external data structure can contain up to 32,000 variables for TBOL system data. All TBOL programs have access to the global external data structure and the variables it contains. The variables in a global external data structure are character string variables like the ones one defines in the data section of a program. The global external variables are named #1 through #32,000. These variables are assigned and controlled by the TBOL database administrator which maintains a file of DEFINE compiler directive statements which assign meaningful names to the global external variables in use. In the preferred embodiment, the MS-DOS file specification for this file can, for example be TBOLLIB/TBOL.SYS. In this regard, the COPY compiler directive is used to copy TBOL.SYS into a source code input stream. Subsequent statements in the program source code can reference the global external system variables by using the meaningful names assigned by the DEFINE statements in this file.

Examples of global external variables are SUS_RETURN_CODE, which is assigned a return code value after the execution of certain TBOL program verb statements; SYS_DATE, which contains the current system date; and SYS_TIME which contains the current system time.

With regard to the TBOL program code section, as noted above, every TBOL program must have a code section. The code section contains the program logic which is composed of one or more procedures. In accordance with this arrangement, a procedure begins with the keyword PROC followed by an equal sign (=)

46

and then the name of the procedure. The body of the procedure is composed of procedure statements, ending with the END_PROC statement. For example:

```
PROC=proc_name statement [statement . . .] END_PROC;
```

where "proc_name" is an identifier; i.e. the name of the procedure, and "statement" is a TBOL procedure statement as described below.

In accordance with the invention, at program execution time, control is given to the first procedure in the program. This is the mainline procedure. From then on, the flow of procedure execution is controlled by the logic contained in the procedures themselves.

Each procedure statement begins with a TBOL keyword called a verb. However, as noted above, the name of a procedure can also act as the verb in a procedure statement, exactly as if it were a TBOL verb. In such case, the data in any statement operand is moved into parameter registers and control is passed to the other procedure. No special linkage or parameter passing conventions are needed. As will be appreciated by those skilled in the art, this is a powerful feature which enables the application programmer to extend the language vocabulary to include his own library of application-oriented verb commands and commonly used procedures.

When control is transferred to another procedure, as noted, the "called" procedure returns control to the "calling" procedure with a RETURN or END_PROC statement, where RETURN and END PROC are TBOL verbs described more fully hereafter. Upon return, the "calling" procedure's parameter data, if any, is restored in the parameter registers, and program execution resumes with the next statement. Recursive logic is possible by using the name of the current procedure as the verb in a procedure statement, thus causing the procedure to "call" itself.

In accordance with the design of TBOL, any procedure statement may be preceded with one or more identifying labels. A label consists of an Identifier followed by a colon (:). For example:

```
(stmt_label: . . .) statement
```

where "stmt_label" is an Identifier, for the statement, and "statement" is a TBOL procedure statement.

Procedure statement labels are used for transferring control to another statement within the same procedure using a GOTO or GOTO_DEPENDING_ON statement (TBOL verbs described more fully hereafter).

GOTO and or GOTO_DEPENDING_ON statement can also be used to transfer control to another procedure. Transfer to another procedure is done by using the target procedure name as the verb in a statement.

Also in accordance with the design of TBOL, all procedural logic is constructed from statements designed to execute in three basic patterns: sequential, conditional, or repetitive. In the case of a sequential pattern, the sequential program logic consists of one or more procedure statements. In the case of a conditional pattern, the conditional program logic is constructed using IF . . . THEN . . . ELSE and GOTO_DEPENDING_ON key words, described more fully hereafter. Finally, in the case of a repetitive pattern, the repetitive program logic is constructed using WHILE . . . THEN key words or IF . . . THEN . . . ELSE and GOTO key words also described more fully hereafter.

In accordance with the TBOL design, a procedure statement may contain operand following the verb. In

the case of procedure statements, there are five types of procedure statement operand; data names; group data names; system registers, label identifiers, and literals. In this arrangement, data names are the names of variables, and data name operand can be either field names; field numbers with subscripts or array names with subscripts. In the case of field names, a field name is the identifier used as the name of a variable in a data structure in the data section of the program, or the name of TBOL-defined variable in an external data structure.

For field names with subscripts, a field name followed by a subscript enclosed in parentheses (()) refers to a following field. The subscript must be an integer number expressed as a literal or contained in a variable field. The subscript base is 1. For example: CUST_NAME(1) refers to the field CUST_NAME, and CUST_NAME(2) refers to the field following CUST_NAME.

For array names with subscripts, an array name is the identifier used as the name of an array in a data structure in the data section of the program. An array name followed by a subscript enclosed in parentheses (()), refers to an individual element in the array. The subscript must be an integer number expressed as a literal or contained in a variable field. The subscript base is 1, so the first element in an array is referenced with a subscript of 1.

In the case of procedure statement group data name operand, the group data names are the names of data structures or arrays. Group data names are used in statements where the verb allows data structures or arrays to be treated as a single unit. For example, the TBOL MOVE verb allows the use of group data name operand. If the names of two arrays as group data operand are used, the contents of each element in the source array is moved to the corresponding element in the destination array. Here the array names are specified without subscripts. However, if the names of two data structures as group data operand are used, the contents of each variable in the source data structure is moved to the corresponding variable in the destination data structure.

With regard to system register operand, they can be either integer registers I1 through I8, or decimal registers D1 through D8, or parameter registers P1 through P8.

In the case of label identifiers, the label identifiers are the identifiers used as procedure statement labels described above.

Continuing, literal operand can be either, integer numbers, decimal numbers or character strings. Where the literal operand are integer numbers, the integer is composed of the digits 0 through 9. Where a negative integer is to be represented, a minus sign (−) is allowed in the left-most position. However, a decimal point is not allowed. Accordingly, the minimum value that can be represented is −32,767, and the maximum value is 32,767. Where the literal operand is a decimal number, the decimal number is composed of the digits 0 through 9 with a decimal point (.) where desired. A minus sign (−) is allowed in the left-most position. Thus the minimum allowable value is −9999999999.9999999999999999, and the maximum value is 999999999999.9999999999999999.

Further, where the literal operand is a character string, the character string is composed of any printable characters or control characters. Character strings are enclosed in single quotes ('). To include a single quote

character in a character strip, it must be preceded with the backslash character (\). For example: \'. To include a new line character in a character string, the control character \n is used. For example: 'this causes a new line:\n'. To include binary data in a character string, the hex representation of the binary data is preceded with the backslash character (\). For example: 'this is binary 01110111:\77'.

The syntax of a complete TBOL program is illustrated in the following example program.

HEADER SECTION	PROGRAM program_name;
DATA SECTION	DATA
:	data_structure_name_1= {1st data structure}
:	.
:	variable_name_1,
:	.
:	variable names
:	.
:	variable_name_n;
:	.
:	data structures
:	.
:	data_structure_name_n= {nth data strctre}
:	.
:	variable_name_1,
:	.
:	variable names followed by commas
:	.
:	variable_name_n;
CODE SECTION	PROC proc_name_1={mainline procedure}
:	.
:	procedure statements
:	.
:	IF x = x THEN EXIT: {if done,ret to:RS Sys}
:	procedure statements
:	.
:	END_PROC; {end of mainline procedure}
:	.
:	procedures
:	.
:	PROC proc_name_n= {nth procedures}
:	.
:	procedure statements
:	.
:	IF x = x THEN RETURN; {if done,ret to: "calling"proc}
:	procedure statements
:	END-PROC; {end of nth procedure}
:	{end of program}

In accordance with the invention, the TBOL compiler enables portability of TBOL programs. Specifically, the TBOL compiler is capable of generating compact data streams from the TBOL source code that can be interpreted by any reception system configured in accordance with the invention, i.e., a personal computer running the reception system application software. For this arrangement, the compiler input file containing the TBOL source code may have any name. For example, the extension .SRC can be used.

During the compilation, three files are generated. Their names are the same as the source code file; their extensions identify their contents. For example, when the file names INPUT.SRC is compiled the following files are generated by the compiler: INPUT.SYM which contains a symTBOL directory; IN-PUT.COD which contains the compiled code; and INPUT.LST which contains the listing.

49

5,347,632

In order to resolve an undefined procedure, the TBOL compiler automatically search the local MS-DOS directory TBOLLIB for a file named procname.LIB, where procname is the name of the unresolved procedure. IF procname.LIB is found, the compiler will automatically copy it into the source code stream after the program source text has ended.

In addition to the undefined procedures facility above noted, the TBOL compiler also may be caused to substitute one text string for another. This accomplished by a DEFINE directive.

Wherever the text pattern specified in operand 1 is found in the source code stream, it is replaced by the compiler with the text pattern specified in operand 2. The syntax for the procedure is:

DEFINE source_pattern, replacement_pattern;
where "source_pattern" is the text in the source code which the compiler is to replace, and "replacement_pattern" is the text the compiler will use to replace source_pattern.

If source_pattern or replacement_pattern contain any blank (space) characters, the text must be enclosed in single quotes (''). Further, the compiler can be made to eliminate certain text from the input source stream by using a null text string for the replacement_pattern ("").

It is to be noted that while DEFINE directives are normally placed in the data section, they can also be placed anywhere in the source code stream. For example, if the name CUST_NUMBER has been used in a TBOL application program to refer to a partition external variable named &6. The DEFINE statement DEFINE CUST_NUMBER,&6 would cause the compiler to substitute &6 whenever it encounters CUST_NUMBER in subsequent statements.

As a further illustration, if the words MAX and MIN are defined with numeric values, DEFINE MAX,1279; and DEFINE MIN,500; MAX and MIN can be used throughout the program source code rather than the actual numeric values. If the values of MAX and MIN change in the future, only the DEFINE statements will need to be changed.

Still further, the compiler can also be caused to copy source code from some other file into the compiler input source code stream. This can be accomplished with a directive entitled COPY. With the use of the COPY directive, the source code contained in the file specified in operand 1 is copied into the source code stream at the point where the COPY statement is read by the compiler. For example, the syntax would be:

COPY 'file_name';

where "file_name" is the name of the file containing source code to be inserted in the source code stream at the point of the COPY statement. In this arrangement, file_name must be enclosed in single quotes (''), and file_name must conform to the operating system file naming rules (in the current preferred embodiment, those of MS-DOS). Further, the file referenced in a COPY statement must reside in the TBOLLIB directory on the compilation machine. In accordance with the invention the COPY statement can be placed anywhere in the source code stream.

By way of illustration, the COPY statement COPY 'TBOL.SYS'; causes the compiler to insert source text from the file TBOL.SYS. This file is maintained by the TBOL Database Administrator, and contains DEFINE statements which assign meaningful names to the TBOL system variables in the global external data structure.

50

As shown in Table 2, 25 verbs are associated with data processing; 15 with program flow; 5 with communications; 6 with file management, 5 with screen management; 1 with object management and 2 with program structure for a total of 59. Following is an alphabetical listing of the TBOL verbs, together with a description of its function and illustration of its syntax.

ADD

The ADD verb adds two numbers. Specifically, the number in operand 1 is added to the number in operand 2. Thus, the number in operand 1 is unchanged, while the number in operand 2 is replaced by the sum of the two numbers. The syntax for ADD is:

ADD number1,number2;

where number1 contains the number to be added to number2. In this arrangement, number1 can be a data name; system register or literal number. As is apparent, number2 contains the second number, and is overlaid with the resulting sum. Number2 can be a data name or system register.

TBOL will automatically perform data conversion when number1 is not the same data type as number2. Sometimes this will result in number2 having a different data type after the add operation. In accordance with this embodiment, fractions will be truncated after 13 decimal places, and whole numbers will not contain a decimal point. Negative results contain a minus sign (-) in the left-most position.

AND

The AND verb performs a logical AND function on the bits of two data fields. The logical product (AND) of the bits of operand 1 and operand 2 is placed in operand 2. Moving from left to right, the AND is applied to the corresponding bits of each field, bit by bit, ending with the last bit of the shorter field. If the corresponding bits are 1 and 1, then the result bit is 1. If the corresponding bits are 1 and 0, or 0 and 1, or 0 and 0, then the result bit is 0. In this arrangement, the data in operand 1 is left unchanged, and the data in operand 2 is replaced by the result.

The AND syntax is:

AND field1,field2;

where "field1" contains the first data field, which can be a data name, system register, I1-I8 or P1-P8 only, or a literal. Continuing, "field2" contains the second data fields, and the contents of field2 are overlaid by the result of the AND operation. Field2 can be a data name, a system register: I1-I8 or P1-P8 only.

As will be appreciated, the AND verb can be used to set a bit to 0.

CLEAR

The CLEAR verb sets one or more variables to null. The CLEAR statement may have either one or two operand. If only one operand is specified, it may contain the name of a field, an array or a data structure. If the operand contains a field name, then that field is set to null. If the operand contains an array name, then all elements of the array are set to null. If the operand contains the name of a data structure, then all fields and array elements in the data structure are set to null. If two operand are specified, then each operand must contain the name of a field. In this case, all fields, beginning with the field in operand 1 and ending with the field in operand 2, are set to null.

The syntax for CLEAR is:

CLEAR name1[,name2];

where "name1" contains the name of a field, array, or data structure to be set to null. If "name2" is specified,

51

name1 must contain a field Lf name. Name1 can be a data name, group data name, or system register P1-P8 only. Further, name2 contains the last field name of a range of fields to be set to null, and can be a data name, group data name, or system register P1-P8 only.

CLOSE

The CLOSE verb is used to close a reception system file after file processing has been completed. By using CLOSE, the file named in operand 1 is closed. If no operand is specified, then all open files are closed. The CLOSE syntax is:

CLOSE [filename];

where, "filename" contains the name of the reception system file to be closed. The file name "PRINTER" specifies the system printer. Otherwise, the name of the file must be a valid MS-DOS file specification; e.g., [drive:][\path \]name[.extension] File name can be a data name, or system register P1-P8 only. When file processing is complete, the file must be closed.

CLOSE_WINDOW

The CLOSE_WINDOW verb is used to close the open window on the base screen and, optionally, open a new window by appending the partial operator _OPEN to the middle of the verb (as shown below). Specifically, by using CLOSE_WINDOW, the open window on the base screen is closed. If no operand is specified, program execution continues with the next statement in the program which last performed an OPEN_WINDOW. If operand 1 is specified, the window whose object ID is contained in operand 1 is opened, and program execution continues with the first statement of the program associated with the newly opened window object.

The CLOSE_WINDOW syntax is:

CLOSE_WINDOW [window-id];

where, "window-id" contains the object ID of a new window to be opened after closing the currently open window. A window-id can be a data name, system register P1-P8 only, or a literal. The CLOSE_WINDOW verb can only be performed by a window program; i.e., a program associated with a window object. CLOSE_WINDOW is the method by which a window program relinquishes control. A window program can also close itself by performing one of the following verbs: NAVIGATE, TRIGGER_FUNCTION. Although a window program cannot perform a OPEN_WINDOW operation, it can use CLOSE_WINDOW to close itself and open another window. This process can continue through several windows. Finally, when a window program performs a CLOSE_WINDOW without opening a new window program control does not work its way back through all the window programs. Instead, control returns to the non-window program which opened the first window. Program execution continues in that program with the statement following the OPEN_WINDOW statement.

CONNECT

The CONNECT verb dials a telephone number. The telephone number contained in operand 1 is dialed. The telephone line status is returned in the system variable SYS_CONNECT_STATUS. The syntax for CONNECT is:

CONNECT phone_number;

where "phone_number" contains the telephone number to be dialed. Phone number can be a data name, system register P1-P8 only, or a literal.

DEFINE_FIELD

5,347,632

52

The DEFINE_FIELD verb is used to define a screen field at program execution time. From five to seven operand specify a single-line or multiple-line field within the currently active screen partition; i.e. the partition associated with the running program. The field is dynamically defined on the current screen partition.

The syntax for DEFINE_FIELD is:

DEFINE_FIELD name,row,coln,width,height [object_id [state]]; where "name" is the field to receive the name of a partition external variable. When this statement is performed, a screen field is defined and it is assigned to a partition external variable. The partition external variable name is placed in the name operand. Name may be a data name, or system register P1-P8 only.

Continuing "row" in the DEFINE_FIELD syntax contains the row number where the field starts. The top row on the screen is row number 1. Row can be a data name, system register P1-P8, or a literal. "Column" contains the column number where the field starts. The left-most column on the screen is column number 1. Column can be a data name, system register P1-P8 only, or a literal. In the DEFINE_FIELD syntax, "width" contains a number specifying how many characters each line the field will hold. Width can be a data name, system register P1-P8 only, or a literal. Further, "height" contains a number specifying how many lines the field will have. For multiple-line fields, each field line will begin in the column number specified in the column operand. Height can be a data name, system register P1-P8 only, or a literal.

Yet further, in the DEFINE_FIELD syntax, "object_id" contains the object ID of a field post processor program that is to be associated with this field. Object_id can be a data name, system register P1-P8 only, or a literal. Finally, for the DEFINE_FIELD syntax "state" contains a character string which is to be placed in parameter register P1 when the program specified in the object_id operand is given control. State can be a data name, system register P1-P8 only or a literal.

In the case of the DEFINE_FIELD verb, if the object-id operand is specified, then the post processor program object is obtained only on a "commit" event; avoiding the need for a synchronous FETCH. Since DEFINE_FIELD defines a field only in the screen partition associated with the running program, a program can not define a field in some other screen partition with which it is not associated. Additionally, page-level processor programs which are not associated with a particular screen partition can not use this verb.

DELETE

DELETE is used to delete a reception system file for file processing. The file named in operand 1 is deleted.

The syntax for DELETE is:

DELETE [filename];

where "filename" contains the name of the reception system file to be deleted. Filename can be a data name or system register P1-P8. Filename must be a valid operating specification.

DISCONNECT

The DISCONNECT verb "hangs up the telephone", thus, terminating the telephone connection. The syntax for DISCONNECT is simply:

DISCONNECT

DIVIDE

The DIVIDE verb divides one number by another. The number in operand 2 is divided by the number in

5,347,632

53

operand 1. The number in operand 1 is unchanged, however, the number in operand 2 is replaced by the quotient. If operand 3 is specified, the remainder is placed in operand 3. The syntax for DIVIDE is:

DIVIDE number1,number2[,remainder];

where "number1" contains the divisor, i.e. the number to be divided into number2. Number1 can be a data name, system register, or literal number. Continuing, "number2" contains the dividend; i.e., the number to be divided by number1. The contents of number2 are overlaid by the resulting quotient. Number2 can be a data name, or a system register. And, "remainder" is a variable or system register designated to hold the remainder of the divide operation. Remainder can be a data name, or a system register.

TBOL will automatically perform data conversion when number1 is not the same data type as number2. Sometimes this will result in number2 having a different data type after the divide operation. Fractions will be truncated after 13 decimal places, while whole number will not contain a decimal point. Negative results will contain a minus sign (-) in the left-most position.

DO . . . END

The keyword DO specifies the beginning of a block of statements; the keyword END specifies the end of the block. A block of statements, bracketed by DO and END can be used as a clause in an IF or WHILE statement. In an IF statement, either the THEN clause or an optional ELSE clause can be executed, based upon the evaluation of a boolean expression. In a WHILE statement, the THEN clause is executed repetitively until a boolean expression is false.

The syntax for DO . . . END is:

DO . . . block . . . END;

where "Block" is any number of TBOL statements. As shown, the keyword DO is not followed by a semicolon, and the END statement requires a terminating semicolon.

EDIT

The EDIT verb gathers and edits data from multiple sources, then joins it together and places it in the specified destination field. Data from one to six sources, beginning with operand 3, is edited in accordance with the mask contained in operand 2. The edited data, joined together as a single character string is places in the output destination field specified in operand 1.

The EDIT syntax is EDIT output,mask, source [source . . .], where "output" contains the name of the destination field for the edited data. After performance of the EDIT statement, the destination field will contain "sub-fields" of data; one for each source operand. Output can be a data name, or a register P1-P8 only.

Continuing, "mask" contains a character string consisting of one edit specification for each source operand. Edit specifications are in the form: %[-][min.max]x, where "%" indicates the beginning of an edit specification; "-" indicates left-adjustment of the source data in the destination sub-field, and "min.max" are two numbers, separated by a decimal point, which specify the minimum and maximum width of the edited data in the destination sub-field, and "x" is an alpha character which controls the retrieval of data from the corresponding source operand. Further, "x" can be a "d" to indicate a digit, characters retrieved from the corresponding source operand are converted to integer format; or "x" can be an "f" to indicate floating point, characters retrieved from the corresponding source operand are converted to a decimal format; or an "x"

54

can be an "s" to indicate a string, characters retrieved from the corresponding source operand are converted to character format; or an "x" can be a "c" to indicate a character, only one character is retrieved from the corresponding source operand, and is converted to character format.

Characters in mask which are not part of edit specifications are placed in output as laterals. Mask can be a data name, or system register P1-P8 only.

Continuous source contains the source data to be edited. The EDIT statement may contain up to six source operand. Mask must contain an edit specification for each source operand specified. Source can be a data name, a system register, or a literal.

END_PROC

The END_PROC verb identifies the last physical statement in a procedure definition. Control returns to the "calling" procedure and program execution continues with the statement following the "call" statement. The syntax for END_PROC is:

END_PROC;

An END_PROC statement is required as the last physical statement in every procedure. Accordingly, a procedure may contain only one END_PROC statement.

An END_PROC statement in a "called" procedure is equivalent to a RETURN statement. Further, an END_PROC statement in the highest level procedure of a program is equivalent to an EXIT statement.

ERROR

The ERROR verb causes the Reception System to reset. Processing resumes with a new page template object. Execution of the currently running program is terminated and control returns to the Reception System. The reception System resets itself. Program execution then resumes with the first statement in the program associated with the page template object specified in operand 1.

The ERROR syntax is:

ERROR object_id;

where "object_id" contains the object ID of a page template object. After the Reception System reset, control is transferred to the program associated with the page template object. Object_id can be a data name, a system register P1-P8 only, or a literal.

The ERROR verb is used to continue a text/graphic session when the currently running program encounters a condition which can only be resolved by a reset of the Reception System.

EXIT

The EXIT verb is used to transfer program control to the Reception System. When EXIT executes, the currently running program is ended. The data in operand 1 is moved to SYS_RETURN_CODE, and control is returned to the Reception System. The syntax for EXIT is:

EXIT return code; where "return-code" contains data to be moved to SYS_RETURN_CODE prior to transfer of control to the Reception System. A value of 0 indicates a normal return. A non-zero value indicates an error condition. Return_code can be a data name, system register, or a literal.

The EXIT verb is the normal way to end processing in a TBOL program. In the highest level procedure of a program a RETURN or an END_PROC is equivalent to an EXIT.

FETCH

55

The **FETCH** verb is used to retrieve an object from a host system or from the Reception System storage device stage. The object specified in operand 1 is retrieved from its present location and made available in the Reception System. If operand 2 is specified, the object's data segment is placed in the operand 2 field.

The syntax for **FETCH** is:

FETCH object_id [, field]; where "object_id" contains the object ID of the object to be located and retrieved. Object_id can be a data name, system register P1-P8 only, or a literal.

In the **FETCH** syntax "field" contains the name of a field to hold the retrieved object's data segment. Field can be a data name, or a system register P1-P8 only.

When an object might be required for subsequent processing, the field operand should not be specified in the **FETCH** statement. In that case, the **FETCH** will be an asynchronous task and the program will not experience a wait. The object is placed in the Reception System ready for use. The field operand is specified when an object is required to immediate use. Here, the **FETCH** is a synchronous task and the program may experience a wait. When the **FETCH** is completed, the program has access to the **FETCHed** object's data segment in the field operand.

FILL

The **FILL** verb is used to duplicate a string of characters repeatedly within a field. The character string pattern contained in operand 2 is duplicated repeatedly in operand 1 until the length of operand 1 is equal to the number specified in operand 3. The syntax for **FILL** is:

FILL output,pattern,length; where "output" is the name of the field to be filled with the character string specified in "pattern". Output can be a data name or a system register P1-P8 only, or a literal. Finally, "length" contains an integer number specifying the final length of output. Length can be a data name, system register or a literal.

FORMAT

The **FORMAT** verb is used to transfer a string of character data into variables defined in the **DATA** section of the program. The string of character data contained in operand 1 is transferred to **DATA** section variables using destination and length specification in the format map contained in operand 2. The **FORMAT** syntax is:

FORMAT source,map;

where "source" contains a string of character data to be transferred to **DATA** section variables, and can be a data name or system register P1-P8 only.

Continuing, "map", on the other hand, contains a format map consisting of a destination/length specification for each field of data to be transferred. Map is created with the **MAKE_FORMAT** verb prior to execution of the statement.

GOTO

The **GOTO** verb transfers control to another statement within the currently running procedure. Program execution continues at the statement with the label identifier specified as operand 1. The syntax for **GOTO** is:

GOTO label_id;

where "label_id" is a label identifier directly preceding a statement within the currently running procedure. A **GOTO** statement can be used to transfer control to another procedure. Transfer to another procedure is accomplished by using the target procedure name as the verb in a statement.

GOTO_DEPENDING_ON

5,347,632

56

The **GOTO_DEPENDING_ON** verb transfers control to one of several other statements within the currently running procedure. Operand 1 contains a number, and is used as an index to select one of the label identifiers beginning with operand 2 in the statement. Program execution continues at the statement with the selected label identifier.

The syntax for **GOTO_DEPENDING_ON** is:

GOTO_DEPENDING_ON index, label_id [, label_id . . .];

where "index" is an integer number used to select one of the label identifiers in the statement as the point where program execution will continue. If index contains a 1, then program execution continues at the statement with the label identifier specified as operand 2. If index contains a 2, then program execution continues at the statement with the label identifier specified as operand 3. And so on. If there is no label_id operand corresponding to the value in index, then program execution continues with the statement following the **GOTO_DEPENDING_ON** statement. Index can be a data name or system register. Continuing, "label_id" is a label identifier directly preceding a statement within the currently running procedure. Up to 147 label_id operands may be specified in a **GOTO_DEPENDING_ON** statement.

A **GOTO_DEPENDING_ON** statement, however, cannot be used to transfer control to another procedure. Transfer to another procedure is done by using the target procedure name as the verb in a statement.

IF . . . THEN . . . ELSE

In this verb, the keyword **IF** directs the flow of program execution to one of two possible paths depending upon the evaluation of a boolean expression. The keyword **IF** is followed by a boolean expression. The boolean expression is always followed by a **THEN** clause. The **THEN** clause may be followed by an **ELSE** clause. The boolean expression is evaluated to determine whether it is "true" or "false". If the expression is true, program execution continues with the **THEN** clause; the **ELSE** clause, if present, is skipped. If the expression is false, the **THEN** clause is skipped; program execution continues with the statement following the clause or clauses.

The syntax for **IF . . . THEN . . . ELSE** is:

IF boolean **THEN** clause [**ELSE** clause];

where "boolean" is a boolean expression. Boolean can be a single relational expression or two or more relational expressions separated by the key words **AND** and **OR**. These relational expressions can be enclosed with parentheses, and then treated as a single relational expression separated from others with **AND** or **OR**. They are evaluated from left to right.

In the syntax, "clause" can be: a single statement, or a block of statements. Where clause is a block of statements, the block begins with the keyword **DO** and ends with the **END** verb. Further, Clause is always preceded by the keyword **THEN** or **ELSE**.

INSTR

The **INSTR** verb searches a character string to determine if a specific substring of characters is contained within it. The character string in operand 1 is searched for the first occurrence of the character string in operand 2. If a matching string is found in operand 1, an integer number specifying its starting position is placed in operand 3. If a matching string is not found, 0 is placed in operand 3.

The syntax for **INSTR** is:

57

INSTR string, pattern strt_pos;
where "string" contains the character string to be searched. String can be a data name, system register P1-P8 only, or a literal.

Continuing, "pattern" contains the character string pattern which may occur within the string operand, and can be a data name, system register P1-P8 only, or a literal.

Finally, "strt_pos" is the name of the variable where the starting position (or o) is to be stored. Strt_pos can be a data name, or system register P1-P8 only.

LENGTH

The LENGTH verb is used to determine the length of a specified variable. An integer number specifying the number of characters in operand 1 is placed in operand 2. The syntax for LENGTH is:

LENGTH field, length;
where "field" contains the data whose length is to be determined. Field can be a data name, system register P1-P8 only, or a literal.

Continuing, on the other hand, "length" is the name of the variable which is to contain the length of the field operand, and can be a data name, or a system register P1-P8 only.

LINK

The LINK verb transfers control to another TBOL program. Program execution continues at the first statement in the program whose object ID is contained in operand 1. Up to eight parameters may be passed to the "called" program in operands 2-9. Control returns to the statement following the LINK statement when the "called" program performs an EXIT.

The syntax for LINK is:

LINK object_id [, parameter . . .];
where "object_id" contains the object ID of a TBOL program, and can be data name, system register P1-P8, only or a literal. Further, "parameter" contains parameter data for the program whose object ID is contained in operand 1. The contents of the parameter operand 2 through 9, if present, are placed in parameter registers P1 through P8. The number of parameter operand is placed in P0. P0 through P8 are accessible to the "called" program. Parameter can be a data name, system register, or a literal.

LOOKUP

The LOOKUP verb issued to search for an entry in a table of data contained in a character string. Operand 2 contains a single character string consisting of a number of logical records of equal length. Each record consists of a fixed-length key field and a fixed-length data field. Operand 3 contains the record length.

Operand 1 contains a search key equal in length to the length of the key field. Operand 2 is searched for a record with a key field equal to operand 1. If a record with a matching key is found, an integer number specifying its starting position is placed in operand 4. If a matching record is not found, 0 is placed in operand 4.

The syntax for LOOKUP is:

LOOKUP schkey,table,rcd_lth,result; where
"schkey" contains the key data of the desired record and can be a data name, system register or a literal. Further, "table" contains a character string consisting of a number of equal length logical records, and be a data name or system register P1-P8 only. Yet further, "rcd_lth" contains an integer number equal to the length of a record in a table, and can be a data name, system register, or a literal. Finally, "result" is the name

5,347,632

58

of the field to receive the result of the search. Result can be a data name, or a system register.

MAKE_FORMAT

The MAKE_FORMAT verb is used to create a format map for use with the FORMAT verb. From 1 to 255 destination/length specifications contained in operand (beginning in operand 2) are used to create a format map which is stored in operand 1. Operand 1 can then be specified as the map operand in a FORMAT statement.

The MAKE_FORMAT syntax is:

MAKE_FORMAT map, format [, format . . .];

where "map" is the name of the variable which is to contain the format map created with this statement. Map will be specified as an operand in a subsequent FORMAT statement to control the transfer of a string of character data to variables. Map can be either a data name or system register P1-P8 only. Continuing, "format" contains a destination/length specification for one logical field of a string of character data. From 1 to 255 format operand can be specified in this statement to create a format map. Each format operand controls the transfer of one logical field of data from a character string when the format map created in this statement is used in a subsequent FORMAT statement. In this arrangement, format can be a data name or a system register P1-P8 only.

A destination/length specification in a format operand always contains a destination field name. The field name is followed by either one or two integer numbers controlling the length of the designation field data. The field name and numbers are separated by the colon character, e.g., destination:fix_lth:imbed_lth, or destination:fix_lth, or as destination::imbed_lth.

For this approach, "destination" is a variable field name which will contain the logical field of data from the character string after the subsequent performance of the FORMAT verb. And, "fix_lth" is an integer number between 1 and 33767 specifying a fixed field length for destination. If fix_lth is not specified then 2 colon characters are used to separate destination from imbed_lth, showing that fix_lth has been omitted. In this case, the destination field length is controlled entirely by imbed_lth, which must be specified. If fix_lth is specified and imbed_lth is not, then fix_lth characters will be transferred to destination during the subsequent performance of the FORMAT verb. Finally, if fix_lth is specified with imbed_lth, then destination will have a length of fix_lth after the transfer of data by the FORMAT verb.

Continuing, "imbed_lth" is an integer number, either 1 or 2 which specifies length of an imbedded length field that immediately precedes the logical field of data in the character string. The imbedded length field contains the length of the logical field of data immediately following. For example, 1 specifies a 1-character length field and 2 specified a 2-character length field.

If imbed_lth is not specified then the designation field length is controlled entirely by fix_lth, which must be specified. If imbed_lth is specified and fix_lth is not, then the number of characters transferred to destination from the character string is controlled by the number in the one or two-character length field which precedes the logical field of data. If imbed_lth is specified with fix_lth, then the number of characters transferred to destination from the character string is controlled by the number in the one or two-character length field which precedes the logical field of data.

5,347,632

59

After the transfer of data, if the length of destination is not equal to fix_lth, then it is either truncated, or extended with blank characters as necessary.

MOVE

The move verb copies data from one or more source fields into an equal number of destination fields. The data contained in the operand 1 data structure field (or fields) replaces the contents of the operand 2 data structure field (or fields). Operand 1 data remains unchanged. Normally, the moved data is converted to the data type of the destination. If the key word ABS is included as operand 3, then data conversion does not take place.

The syntax for MOVE is:

MOVE source,destination[, ABS];

where "source" is the name of the data structure containing the data to be moved, and can be a data name, or a group data name, or system register, or a literal. Further "destination" is the name of the data structure field (or fields) to receive the source data, and can be a data name, or group data name, or a system register. Finally, "ABS" is a keyword specifying an absolute move; i.e., no data conversion takes place. However, data residing in an integer register will always be in binary integer; and data residing in a decimal register will always be in internal decimal format.

If the source operand is a group data name, then the destination operand must be a group data name. Further, data in all of the fields contained in the source data structure or array are moved to the corresponding fields in the destination data structure or array.

MULTIPLY

The MULTIPLY verb multiplies two numbers. The number in operand 2 is multiplied by the number in operand 1. The number in operand 1 is unchanged. The number in operand 2 is replaced with the product of the two numbers. The syntax for MULTIPLY is:

MULTIPLY number1, number2;

where "number1" contains the first number factor for the multiply operation, and can be a data name, system register or literal; and "number2" contains the second number factor for the multiply operation. Following execution, the contents of number2 are overlaid with the resulting of the product. Number2 can be a data name, or a system register.

TBOL will automatically perform data conversion when number1 is not the same data type as number2. Sometimes this will result in number2 having a different data type after the add operation. Fractions will be truncated after 13 decimal places, and whole numbers will not contain a decimal point. Negative results will contain a minus sign (–) in the left-most position.

NAVIGATE

The NAVIGATE verb is used to transfer control to the TBOL program logic associated with different page template objects. The external effect is the display of a new screen page. Operand 1 contains either a page template object ID, or a keyword representing a navigation target page. Control is returned to the Reception System where the necessary objects are acquired and made ready to continue the videotext session at the specified new page.

The syntax for NAVIGATE is:

NAVIGATE object_id;

where "object_id" contains the object ID of a target page template object, and can be a data name, register P1–P8 only, or a literal.

NOTE

60

The NOTE verb returns the current position of the file pointer in a reception system file. Operand 1 contains the name of a file. An integer number specifying the current position of the file's pointer is returned in operand 2. The NOTE syntax is:

NOTE filename,position;

where "filename" contains the name of a reception system file. The name of the file must be a valid MS-DOS file specification; e.g. [drive:][\path\]name[.extension]. Filename can be a data name, or a system register P1–P8 only. Continuing, "position" is the name of the field to receive the current position of the file pointer for the file specified in filename. This will be an integer number equal to the numeric offset from the beginning of the file; a 10 in position means the file pointer is positioned at the 10th character position in the file. Position can be a data name, or system register.

OPEN

The OPEN verb is used to open a reception system file for file processing. The file named in operand 1 is opened for processing in the mode specified in operand 2. The syntax for OPEN is:

OPEN filename, INPUT:OUTPUT:I/O:APPEND: :BINARY; where

"filename" contains the name of the reception system file to be opened. As will be appreciated with this convention, the file name PRINTER specified the system printer. Otherwise, the name of the file must be a valid MS-DOS file specification; e.g. [drive:][\path\]name[.extension]. File name can be a data name, or system register P1–P8 only.

Further, "INPUT" is a keyword specifying that the file is to be opened for reading only; "OUTPUT" is a keyword specifying that the file is to be opened for writing only; "I/O" is a key word specifying that the file is to be opened for both reading and writing; "APPEND" is a keyword specifying that the file is to be opened for writing, where new data is appended to existing data; and "BINARY" is a keyword specifying that the file is to be opened for both reading and writing. Where all file data is in binary format.

OPEN_WINDOW

The OPEN_WINDOW verb is used to open a window on the base screen. The window whose object ID is contained in operand 1 is opened. Program execution continues with the first statement of the program associated with the newly opened window object. The syntax for OPEN_WINDOW is:

OPEN_WINDOW window_id;

where "window_id" contains the object ID of the window to be opened on the base screen, and can be a data name or system register P1–P8 only or a literal.

After performance of the OPEN_WINDOW statement, program execution continues with the first statement of the window program; i.e., the program associated with the newly opened window object. A window program relinquishes control by performing a CLOSE_WINDOW. Although a window program cannot perform an OPEN_WINDOW, it can use CLOSE_WINDOW to close itself and open another window. This process can continue through several windows. Finally, when a window program performs a CLOSE_WINDOW without opening a new window, program control does not work its way back through all the window programs. Instead, control returns to the non-window program which opened the first window. Program execution continues in that program with the statement following the OPEN_WINDOW statement.

5,347,632

61

A window program can also close itself by performing one of the following verbs: NAVIGATE; or TRIGGER_FUNCTION. In such cases, control does not return to the program which opened the window.

OR

The OR verb performs a logical OR function on the bits of two data fields. The logical sum (OR) of the bits of operand 1 and operand 2 is placed in operand 2. Moving from left to right, the OR is applied to the corresponding bits of each field, bit by bit, ending with the last bit of the shorter field.

If the corresponding bits are 1 and 1, then the result bit is 1. If the corresponding bits are 1 and 0, or 0 and 1, then the result bit is 1. If the corresponding bits are 0 and 0, then the result bit is 0.

The data in operand 1 is left unchanged. The data in operand 2 is replaced by the result.

The syntax for OR is:

OR field1,field2;

where "field1" contains the first data field, and can be a data name, or system register I1-I8 or P1-P8 only, or a literal. Further, "field2" contains the second data field. The contents of field2 are overlaid by the result of the OR operation. Field2 can be a data name, or system register I1-I8 or P1-P8 only. As will be appreciated by those skilled in the art, the OR verb can be used to set a bit to 1.

POINT

The POINT verb is used to set the file pointer to a specified position in a reception system file. Operand 1 contains the name of a file. The file's pointer is set to the position specified by the integer number in operand 2. The POINT syntax is:

POINT filename,position;

where "filename" contains the name of a reception system file. The name of the file must be a valid MS-DOS file specification; e.g. [drive:][\path\]name[.extension]. File name can be a data name, or system register P1-P8 only. Further, "position" contains an integer number equal to the desired position of the file pointer for the file specified in filename. A 10 in position means the file pointer will be positioned at the 10th character position in the file. Position can be a data name, or system register or literal.

POP

The POP verb transfers data from the top of the system stack to a variable field. The contents of operand 1 are replaced with data removed from the top of the system stack. The POP syntax is:

POP field;

where "field" is the name of the variable field to receive data from the stack, and can be a data name, or a system register.

PUSH

The PUSH verb transfers data from a variable field to the top of the system stack. The data contained in operand 1 is placed on the top of the system stack, "pushing down" the current contents of the stack. The contents of operand 1 remain unchanged. The PUSH syntax is:

PUSH field;

where "field" is the name of the variable field containing data to be "pushed" on the stack, and can be a data name, or a system register, or a literal.

READ

The READ verb is used to read data from a reception system file into a variable field. Operand 1 contains the name of a file. Data is read from the file, beginning with the character position specified by the current contents

62

of the file's pointer. Data read from the file replaces the contents of operand 2. Operand 3 may be present, containing an integer number specifying the number of characters to be read. For ASCII files, data is read from the file until the first end-of-line character (ASCII 13) is encountered. Or, if operand 3 is present, until the number of characters specified in operand 3 is read. For binary files, operand 3 is required to specify the length of the data to be read from the file.

The syntax for READ is:

READ filename, input [,length];

where "filename" contains the name of a reception system file, which must be a valid MS-DOS file specification; e.g. [drive:][\path\]name[.extension]. File name can be a data name, or system register P1-P8 only. Continuing, "input" is the name of the variable field to receive data read from the file, and can be a data name, or a system register P1-P8 only. Finally, "length" contains an integer number. For ASCII files, length specifies the maximum number of characters to be read. For binary files, length specifies the length of the data to be read.

As will be appreciated by those skilled in the art, in order to perform a READ operation, a file must first be opened as INPUT or I/O before the READ operation can take place.

RECEIVE

The RECEIVE verb is used to access the expected reply to a message sent previously to a host system. Operand 1 contains the message ID of a message sent previously to a host system. The message reply from the host replaces the contents of operand 2. The RECEIVE syntax is:

RECEIVE msg_id,message;

where "msg_id" contains the message ID of a message sent previously to a host system, and can be a data name, or a system register P1-P8 only. Further, "message" is the name of the variable field to receive the incoming message reply, and can be a data name, or a system register P1-P8 only.

RELEASE

The RELEASE verb reclaims memory space in the reception system by deleting a block of data saved previously with the SAVE verb. The block

of data named in operand 1 is deleted from memory.

The syntax for RELEASE is:

RELEASE block_name;

where "block_name" contains block name used in some previously performed SAVE statement, and can be a literal.

REFRESH

The REFRESH verb causes the current screen fields to receive the contents of the associated partition external variables. The contents of all fields on the current screen are replaced with the contents of their corresponding partition external variables. The REFRESH syntax is:

REFRESH.

The REFRESH operation occurs automatically whenever all programs for a given event (for example, commit; field end; or initial display) have finished execution. Therefore, a program should only perform a REFRESH statement if fields are updated during an event.

RESTORE

The RESTORE verb is used to restore the previously saved contents of a block of variables. The block of data named in operand 1 replaces the contents of a block of

63

variables, beginning with the variable named in operand 2. The RESTORE syntax is:

RESTORE block_name,field1;

where "block_name" contains a block name used in some previously performed SAVE statement, and can be a literal. Further, "field1" is the name of the first field or a data structure to receive data from the block specified in block_name. Field1 can be a data name, or a group data name.

RETURN

The RETURN verb is used to return control to the procedure which "called" the currently running procedure. Execution of the currently running procedure is ended. The data in operand 1 is moved to SYS_RETURN_CODE, and control is returned to the procedure which "called" the currently running procedure.

The RETURN syntax is:

RETURN return-code;

where "return-code" contains data to be moved to SYS_RETURN_CODE prior to transfer of control to the "calling" procedure, and can be a data name, or system register, or a literal. It should be noted that in the highest level procedure of a program, a RETURN or an END_PROC is equivalent to an EXIT.

SAVE

The SAVE verb is used to save the contents of a block of variables. Operand 1 contains a name to be assigned to the block of saved data. This name will be used later to restore the data. If operand 2 is specified without operand 3, then operand 2 may contain the name of a field, an array, or a data structure. In this case, the contents of the field; or the contents of all the elements in the array; or the contents of all the fields in the data structure are saved under the name specified in operand 1. If operand 2 and operand 3 are specified, then they both must contain a field name. In this case, the contents of all the fields, beginning with the field in operand 1 and ending with the field in operand 2, are saved under the name specified in operand 1.

The syntax for SAVE is:

SAVE block_name,name1[,name2];

where "block_name" contains a block name to be assigned to the saved data, and will be used subsequently to restore the saved contents of the fields. Block_name can be a data name, system register P1-P8 only, or a literal. Continuing, "name1" contains the name of a field, array, or data structure to be saved. If name2 is specified, name1 must contain a field name. Name1 can be a data name. Further, "name2" contains the last field name of a range of fields to be saved, and it can be a data name.

SEND

The SEND verb is used to transmit a message to a host system. The message text contained in operand 2 is transmitted from the reception system using a message header constructed from the data contained in operand 2. Operand 3, if present, indicates that an incoming response to the message is expected. The syntax for SEND is:

SEND message [,RESPONSE:TIMEOUT];

where "message" contains the outgoing message text (the header data for which has been placed in GEVs before SEND), and can be a data name, or a system register, or a literal. "RESPONSE" is a keyword indicating that a response to the message is expected. "TIMEOUT" is a parameter that sets the number of seconds for message time-out.

5,347,632

64

After performance of the SEND statement, the global external system variable SYS_LAST_MSG_ID contains a message ID number assigned to the outgoing message by the Reception System. This message ID number can be used later in a RECEIVE statement.

SET_ATTRIBUTE

The SET_ATTRIBUTE verb is used to set or change the color and input format attributes of a screen field. The characteristics of the screen field expressed as operand 1 are set or changed according to the specifications contained in operand 2. The syntax for SET_ATTRIBUTE is:

SET_ATTRIBUTE name, attr_list;

where "name" expresses the name of the field whose characteristics are to be set or changed. This is a partition external variable name, and if the name is expressed as a literal; e.g., "SET_ATTRIBUTE 1, . . .", then this is taken to mean that the attributes of the partition external variable &1 contains the name of the partition external variable whose attributes are to be set by this statement.

Further, "attr_list" is a literal character string containing a list of key words and values describing the desired attributes to be assigned to the field expressed in operand 1.

When SET_ATTRIBUTE is performed, existing field attributes remain in effect unless superseded by the attribute list contained in operand 2. The attribute list operand literal is in the form:

keyword[(values)],keyword[(values)]. . .].

It should also be noted that where key words and their associated values are: "DISPLAY", not user input data can be entered in a field with this attribute; "INPUT", a field with this attribute can receive user input data; "ALPHABETIC", an INPUT field with this attribute can receive any alphabetic character: A through Z, and blank; "ALPHANUMERIC", an "INPUT", field with this attribute can receive any displayable character; "NUMERIC", an INPUT field with this attribute can receive any numeric character: 0 through 9, (\$), (,), (.), and (-); "PASSWORD", an INPUT field with this attribute is intended for use as a password field. Any character entered by the user is displayed in the field as an asterisk (*); "ACTION", a field with this attribute is a TBOL "action" field; "COLOR-(fg,bg)", where fg and bg are numeric values specifying the foreground and background colors of the field; "FORM-(pattern)", where pattern specifies the input data format for this field. Pattern may contain "A", an alphabetic character of A through Z, which must be in this position; "a", an alphabetic character of A through Z, or a blank, which must be in this position; "N" a numeric character of 0 through 9, or (\$), (,), (.), or (-) which must be in this position; "n", a numeric character of 0 through 9, or (\$), (,), (.), (-), or a blank may occupy this position; "X", any displayable character which must be in this position; and "x", any displayable character or a blank which must be in this position.

Any other character in the pattern is displayed in the field as a literal, and acts as an autoskip character at user input time. To include any of the pattern characters as literals in the pattern, they must be preceded by the backslash character. For example, to include the character "A: as a literal in a pattern it would code as "\A". To include the backslash character as a literal, it would code as "\\\".

SET_CURSOR

5,347,632

65

The SET_CURSOR verb moves the cursor to the field specified as operand 1, itself specified as a field number. The syntax for the SET_CURSOR verb is:

SET_CURSOR [field number]

SET_FUNCTION

The SET_FUNCTION verb changes and/or filters a "logical function" process program. The syntax for SET_FUNCTION is: SET_FUNCTION function_id, status[,program_object_id [,state]]; where "function_id" is the logical function" identifier; "status" is one of the following key words: "DISABLE"; "FILTER"; or "ENABLE". DISABLE is used to deactivate "logical function". FILTER is used to execute the logic contained in program_object_id prior to executing the normal "logical function" process. It the logic contained in program_object_id returns a non-zero SYS_RETURN_CODE < the normal "logical function" process will not execute, otherwise, it begins. ENABLE is used to set "logical function" to normal default process.

Continuing, in the SET_FUNCTION syntax, "program_object_id" is the 13 byte object_id of the TBOL program, (conditional); and "state" is data to be passed to the "logical function" program. The data will reside in the P1 register when logic is executed, (optional).

SORT

The SORT verb is used to sort a range of variable fields into the sequence of the key contained in each field. Each variable field contains a record consisting of a fixed-length key field followed by a data field. The key field is the same length in each record. Operand 1 contains the name of the first field in the range of fields to be sorted; operand 2 contains the name of the last field. Operand 3 contains an integer number specifying the length of the key field contained in the beginning of each field. The fields in the range specified by operand 1 and operand 2 are sorted into the sequence of the key field.

The syntax for SORT is:

SORT field1, field2, key_lath;

where "field1" contains the first field name of the range of fields to be sorted, and can be a data name, or system register P1-P8 only; "field2" contains the last field name of the range of fields to be sorted and can be a data name; or system register P1-P8 only; and "key_lath" contains an integer number equal to the length of the key field contained in each field in the range. Key_lath can be a data name, or system register P1-P8 only or a literal.

SOUND

The SOUND verb is used to produce a sound through the reception system speaker. A sound is produced of the pitch specified by operand 1, for the duration specified by operand 2. If operand 1 and operand 2 are not present, values from the most recently performed SOUND statement are used. The SOUND syntax is:

SOUND [pitch,duration];

where "pitch" is a numeric value in the range of 0 to 20,000 specifying the desired pitch of the sound. Pitch can be a data name, system register P1-P8, or a literal; and "duration" is a numeric value in the range of 0 to 65,535 specifying the desired duration of the sound in increments of 0.1 seconds. Duration can be a data name, or system register P1-P8 only or literal.

STRING

The STRING verb joins multiple character strings together with into one character string. Up to eight

66

character strings, beginning with the character string contained in operand 1, are joined together sequentially. The resulting new character string replaces the contents of operand 1. The STRING syntax is:

5 STRING string1, [string . . .];

where "string1" is empty, or contains the character string which will become the left-most portion of the new character string, and a data name, or a system register P1-P8 only; "string" is empty, or contains the character string to be joined behind the character strings in preceding operand, and can be a data name, or system register P1-P8 only or a literal.

SUBSTR

The SUBSTR verb is used to copy a substring of 15 characters from a character string into a designated variable field. The character string containing the substring is in operand 1. Operand 3 contains an integer number equal to the position of the first character to be copied. Operand 4 contains an integer number equal to the number of characters to be copied. The specified substring is copied from the character string in operand 1 and replaces the contents of operand 2.

The syntax for SUBSTR is:

25 SUBSTR string, destination, strt_pos, length; where "string" contains a character string, and can be a data name or system register P1-P8 only, or a literal; "destination" is the name of the variable field to receive the substring copied from the string operand, and can be a data name, or system register P1-P8 only, "strt_pos" contains an integer number specifying the position of the first character to be copied into the destination operand, and can be a data name, or system register or a literal; and "length" contains an integer number specifying the number of characters to be copied into the destination operand, and can be a data name, or system register or a literal.

In accordance with this arrangement, the SUBSTR operation does not take place if: if the length operand is 0, or if the Strt_/DOS operand is greater than the 40 length of the string operand.

SUBTRACT

The SUBTRACT verb subtracts one number from another. The number in operand 1 is subtracted from the number in operand 2. The number in operand 1 is unchanged. The number in operand 2 is replaced by the arithmetic difference between the two numbers. The 45 syntax for SUBTRACT is:

SUBTRACT number1, number2;

where "number1" contains the number to be subtracted 50 from number2, and can be a data name, or system register, or a literal; "number2" contains the second number. As noted, the contents of number2 are overlaid with the resulting difference. Number2 can be a data name, or system register.

TBOL will automatically perform data conversion when number1 is not the same data type as number2. Sometimes this will result in number2 having a different data type after the subtract operation. Fractions will be truncated after 13 decimal places, and whole numbers will not contain a decimal point. Further, negative results will contain a minus sign (-) in the left-most position.

TRANSFER

The TRANSFER verb transfers control to another 65 TBOL program. Control however, does not return to the original program. Rather, program execution continues at the first statement in the program whose object ID is contained in operand 1. Up to eight parameters

67

may be passed to the "called" program in operand 2-9. Control is transferred to the Reception System when the "called" program performs an EXIT.

The syntax for TRANSFER is:

TRANSFER object_id [,parameter...];

where "object_id" contains the object ID of a TBOL program, and can be a data name, or system register P1-P8 only, or a literal; "parameter" contains parameter data for the program whose object ID is contained in operand 1. The contents of the parameter operand 2 through 9, if present, are placed in parameter registers P1 through P8. The number of parameter operand is placed in P0. P0 through P8 are accessible to the "called" program. Parameter can be a data name, or system register, or a literal.

TRIGGER_FUNCTION

The TRIGGER_FUNCTION verb is designed to execute a "logical function". Its syntax is:

TRIGGER_FUNCTION function_id;

where "function_id" is the logical function" identifier. In accordance with the design of TRIGGER_FUNCTION, control may or may not be returned depending on the function requested.

UPPERCASE

The UPPERCASE verb converts lowercase alphabetic characters to uppercase alphabetic characters. Lowercase alphabetic characters (a-z) in the character string contained in operand 1 are converted to uppercase alphabetic characters (A-Z). The syntax for UPPERCASE is:

UPPERCASE string;

where "string" contains a character string, and can be a data name, or a system register P1-P8 only.

WAIT

The WAIT verb causes program control to be given to the REception System for the number of seconds defined in the parameter head. Control is given to the Reception System for one "time slice" and then returned to the currently running program.

The WAIT syntax is simply:

WAIT;seconds

WHILE . . . THEN

The key word WHEN causes a single statement or a block of statements to be executed repetitively while a specified boolean expression is true. The key word WHILE is followed by a boolean expression. The boolean expression is always followed by a THEN clause. The boolean expression is evaluated to determine whether it is "true" or "false". If the expression is true, the THEN clause is executed and the expression is evaluated again. If the expression is false, program execution continues with the statement following the THEN clause.

The syntax for WHILE . . . THEN is:

WHILE boolean THEN clause;

where "boolean" is a boolean expression, which can be a single relational expression, where a relational expression consists of two operands separated by a relational operator such as (=), (<>), (<), (>), (<=), or (>=), or two or more relational expressions separated by the key words AND or OR. These relational expressions can be enclosed with parentheses, and then treated as a single relational expression separated from others with and or OR. Further, they are evaluated from left to right. Continuing, with the syntax for WHILE . . . THEN, "clause" can be either a single statement, a block of statements, where the block begins with the key word GO and ends with the END verb.

5,347,632

68

When character strings of unequal length are compared lexicographically, the longer string is truncated to the length of the shorter string before the comparison. If the shorter string compares "high", then the longer string is "lower". For example: When comparing "GG" to "H", "GG" is valued as less than "H". If the shorter string compares "low" or "equal", then the longer string is "high". For example: When comparing "TO" to "TOO", "TO" is less than "TOO".

In this regard, truncation is done outside of the operand, which the operand remaining the same length after the evaluation.

WRITE

WRITE is the verb used to write records to a file. The syntax for WRITE is:

WRITE filename, output_area [, key]; where "filename" is the name of the file that the record is to be written to, and can be a field_id, array_id(subscript), partition_external_id, global_external_id, or a literal; "output_area" is the name of the area from which the record will be created, and can be a field_id, array_id(subscript), partition_external_id or a global_external_id; and "length" specifies either the maximum number of characters to be read from an ASCII file, or the length of data to be read from a binary file. The file must have been previously opened as OUTPUT, APPEND, or I/O.

The XOR verb performs a logical XOR function on the bits of two data fields. The modulo-two sum (exclusive OR) of the bits of operand 1 and operand 2 is placed in operand 2. Moving from left to right, the XOR is applied to the corresponding bits of each field, bit by bit, ending with the last bit of the shorter field. If the corresponding bits are 1 and 0, or 0 and 1, then the result bit is 1. If the corresponding bits are 1 and 1, or 0 and 0, then the result bit is 0. The data in operand 1 is left unchanged. The data in operand 2 is replaced by the result.

The syntax for XOR is:

XOR field1,field2;

where "field1" contains the first data field, and can be a data name, a system register I1-I8 or P1-P8 only, or a literal; and "field2" contains the second data field. As in other logic operations, the contents of field2 are overlaid by the result of the XOR operation. Field2 can be a data name, system register I1-I8 or P1-P8 only.

As will be appreciated by those skilled in the art, the XOR verb can be used to invert a bit. Further, any field XOR'ed with itself becomes all zeros, and, the sequence: XOR A.B; XOR B.A; XOR A.B; causes the contents of A and B to be exchanged.

GLOBAL EXTERNAL SYSTEM VARIABLES

In accordance with the design of TBOL, names have been assigned to the TBOL system variables in the global external variables (GEV) data structure. The names of GEVs are assigned in DEFINE statements as described above and in the file TBOL.SYS. There are a total of 32,000 GEVs. The first 256 GEVs are reserved for the system, and the remaining 31,744 are assigned as application variables, and are application specific. Since system variables referenced by TBOL interpreter as global variables and are ASCII strings, a system variable table is constructed so that reception system native code can access them as binary integer. An adaptation of this table from the source code file "\rs\rsk\c\sysvar.c", presented in more detail hereafter, is shown in Table. 1.

TABLE 1

SYSTEM GLOBAL EXTERNAL VARIABLES	
System Variable Name	GEV# Description
Sys_rtn_code;	0001 API instr. return code.
Sys_api_event;	0002 API event: post,pre,init or sel
Sys_logical_key;	0003 Current logical key.
Sys_last_msg_id;	0004 Last message id.
Sys_tone_pulse;	0005 Phone type pulse/tone.
Sys_line_status;	0006 Line connection status.
Sys_keyword;	0007 Keyword flag.
Sys_automatic_uppercase;	0008 Auto uppercase.
Sys_scroll_increment;	0009 Scroll increment.
Sys_current_field;	0010 Current field.
Sys_date;	0011 system date.
Sys_time;	0012 system time.
Sys_current_page;	0013 current page.
Sys_selected_obj_id;	0014 sel object id.
Sys_navigate_obj_id;	0015 nav object id.
Sys_cursor_row;	0016 cursor row position.
Sys_cursor_col;	0017 cursor col position.
Sys_path;	0018 user personal path table.
Sys_ttx_phone;	0019 dial trintex phone #.
Sys_total_pages;	0020 total pages in page set.
Sys_page_number;	0021 curr. page of of n pages.
Sys_base_obj_id;	0022 curr. base page object-id.
Sys_window_id;	0023 curr. window object-id.
Sys_path_ptr;	0024 curr. path location.
Sys_keywords;	0025 keyword list.
Sys_current_cursor_pos;	0026 curr. cursor position.
Sys_current_background_color;	0027 curr background color.
Sys_current_foreground_color;	0028 curr foreground color.
Sys_hardware_status;	0029 nature of hard error.
Sys_nocomm;	0030 send:don't send to S1.
Sys_um_dia_header;	0031 header unsolicited msg.
Sys_um_message_text;	0032 text unsolicited msg.
Sys_ca_error_track_info;	0033 error tracking data.
Sys_assisant_current_info;	0034 curr. context info.
Sys_screen_data_table;	0035 data table copy & file.
Sys_ad_list;	0036 pointer to AD list.
Sys_current_keyword;	0037 pointer to cur. keyword.
Sys_previous_keyword;	0038 pointer to prev. keyword.
Sys_guide;	0039 guide.
Sys_previous_menu;	0040 prev menu object-id.
Sys_previous_seen_menu;	0041 prev seen menu obj-id.
Sys_scan_list;	0042 pointer to scan list.
Sys_scan_list_pointer;	0043 user scan list pointer.
Sys_path_name;	0044 Pointer to path name.
Sys_navigate_keyword;	0045 Navigate to keyword.
Sys_keyword_table;	0046
Sys_keyword_disp;	0047
Sys_keyword_table_entry_length;	0048
Sys_keyword_length;	0049
Sys_ext_table;	0050
Sys_data_collect;	0051 Indicates Tracking status.
Sys_fm0_txhdr;	0052 DIA message header
Sys_fm0_txdid;	0053
Sys_fm0_txrid;	0054
Sys_fm4_txhdr;	0055
Sys_fm4_txuseid;	0056
Sys_fm4_txcorid;	0057
Sys_fm64_txhdr;	0058
Sys_fm64_txdata;	0059
Sys_fm0_rxhdr;	0060
Sys_fm4_rxhdr;	0061
Sys_fm4_rxuseid;	0062
Sys_fm4_rxcorid;	0063
Sys_fm64_rxhdr;	0064
Sys_fm64_rxdata;	0065
Sys_surrogate;	0066 md
Sys_leave;	0067 md
Sys_return;	0068 md
Sys_int_regs;	0069 md,area for int save stack
Sys_ttx_help_id;	0070 md,id of sys help window/
Sys_selector_data;	0071 md
Sys_selector_path;	0072 md
Sys_logical_event;	0073 am
Sys_user_id;	0074 mg/
Sys_help_appl;	0075 md/
Sys_help_hub_appl_pto;	0076 md/
Sys_access_key_obj_id;	0077 lw,bi/
Sys_word_wrap=1;	0078
Sys_messaging_status;	0079
Sys_version;	0080

TABLE 1-continued

SYSTEM GLOBAL EXTERNAL VARIABLES		
System Variable Name	GEV# Description	
Sys_leader_ad_id;	0081	
Sys_baud_rate;	0082	
Sys_com_port;	0083	
Sys_obj_header;	0084	
Sys_session_status;	0085	
Systbl sys_var_table [] =	NA Define system var table.	
&Sys_rtn_code,	INTLEN,	SYS_INT_TYPE,
&Sys_api_event,	INTLEN,	SYS_INT_TYPE,
&Sys_logical_key,	INTLEN,	SYS_INT_TYPE,
&Sys_last_msg_id,	INTLEN,	SYS_INT_TYPE,
&Sys_tone_pulse,	INTLEN,	SYS_INT_TYPE,
&Sys_line_status,	INTLEN,	SYS_INT_TYPE,
&Sys_keyword,	INTLEN,	SYS_INT_TYPE,
&Sys_automatic_uppercase,	INTLEN,	SYS_INT_TYPE,
&Sys_scroll_increment,	INTLEN,	SYS_INT_TYPE,
&Sys_current_field,	INTLEN,	SYS_INT_TYPE,
&(unsigned int)Sys_date,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_time,	0,	SYS_STR_TYPE,
&Sys_current_page,	0,	SYS_INT_TYPE,
&(unsigned int)Sys_selected_obj_id,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_navigate_obj_id,	0,	SYS_STR_TYPE,
&Sys_cursor_row,	0,	SYS_INT_TYPE,
&Sys_cursor_col,	0,	SYS_INT_TYPE,
&(unsigned int)Sys_path,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_ttx_phone,	0,	SYS_STR_TYPE,
&Sys_total_pages,	INTLEN,	SYS_INT_TYPE,
&Sys_page_number,	INTLEN,	SYS_INT_TYPE,
&(unsigned int)Sys_base_obj_id,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_window_id,	0,	SYS_STR_TYPE,
&Sys_path_ptr,	INTLEN,	SYS_INT_TYPE,
&(unsigned int)Sys_keywords,	0,	SYS_STR_TYPE,
&Sys_current_cursor_pos,	INTLEN,	SYS_INT_TYPE,
&Sys_current_background_color,	INTLEN,	SYS_INT_TYPE,
&Sys_current_foreground_color,	INTLEN,	SYS_INT_TYPE,
&Sys_hardware_status,	INTLEN,	SYS_INT_TYPE,
&Sys_nocomm,	INTLEN,	SYS_INT_TYPE,
&(unsigned int)Sys_um_dia_header,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_um_message_text,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_ca_error_track_info,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_assisant_current_info,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_screen_data_table,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_ad_list,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_current_keyword,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_previous_keyword,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_guide,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_previous_menu,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_previous_seen_menu,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_scan_list,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_scan_list_pointer,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_path_name,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_navigate_keyword,	0,	SYS_STR_TYPE,
&(unsigned int)Sys_keyword_table,	0,	SYS_STR_TYPE,
&Sys_keyword_disp,	INTLEN,	SYS_INT_TYPE,
&Sys_keyword_table_entry_length,	INTLEN,	SYS_INT_TYPE,
&Sys_keyword_length,	INTLEN,	SYS_INT_TYPE,
&(unsigned int)Sys_ext_table,	0,	SYS_STR_TYPE,
&()Sys_data_collect,		
&(unsigned int) Sys_fm0_txhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm0_txdid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm0_txrid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_txhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_txuseid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_txcorid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm64_txhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm64_txdata,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm0_rxhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_rxhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_rxuseid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm4_rxcorid,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm64_rxhdr,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_fm64_rxdata,	0,	SYS_STR_TYPE,
&Sys_surrogate,	INTLEN,	SYS_INT_TYPE,
&(unsigned int) Sys_leave,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_return,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_int regs,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_ttx_help_id,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_selector_data,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_selector_path,	0,	SYS_STR_TYPE,
&Sys_logical_event,	INTLEN,	SYS_INT_TYPE,
&(unsigned int) Sys_user_id,	0,	SYS_STR_TYPE,

TABLE 1-continued

SYSTEM GLOBAL EXTERNAL VARIABLES		
System Variable Name	GEV#	Description
&Sys_help_appl,	INTLEN,	SYS_INT_TYPE,
&(unsigned int) Sys_help_hub_appl_pto,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_access_key_obj_id,	0,	SYS_STR_TYPE,
&Sys_word_wrap,	1,	SYS_INT_TYPE,
&(unsigned int) Sys_messaging_status,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_version,	0,	SYS_STR_TYPE,
&(unsigned int) Sys_leader_ad_id,	0,	SYS_STR_TYPE,
&Sys_baud_rate,	INTLEN,	SYS_INT_TYPE,
&Sys_com_port,	INTLEN,	SYS_INT_TYPE,
&Sys_obj_header,	0,	SYS_STR_TYPE,/RDC
&Sys_session_status,	INTLEN,	SYS_INT_TYPE,

TABLE 2

TBOL VERBS BY FUNCTIONAL CATEGORY		
<u>DATA PROCESSING</u>		
ADD	LOOKUP	SAVE
AND	MAKE_FORMAT	SORT
CLEAR	MOVE	STRING
DIVIDE	MULTIPLY	SUBSTR
EDIT	OR	SUBTRACT
FILL	POP	UPPERCASE
FORMAT	PUSH	XOR
INSTR	RELEASE	
LENGTH	RESTORE	
<u>PROGRAM FLOW</u>		
CLOSE_WINDOW	LINK	TRANSFER
EXIT	NAVIGATE	TRIGGER_FUNCTION
GOTO	OPEN_WINDOW	WAIT
GOTO_DEPENDENT_ON	RETURN	WHILE ... THEN
IF ... THEN ... ELSE	SET_FUNCTION	SYNC_RELEASE
<u>COMMUNICATIONS</u>		
CONNECT	RECEIVE	
DELETE	SEND	
DISCONNECT		
<u>FILE MANAGEMENT</u>		
CLOSE	OPEN	READ
NOTE	POINT	WRITE
<u>SCREEN MANAGEMENT</u>		
DEFINE_FIELD	SOUND	
SET_ATTRIBUTE	REFRESH	
SET_CURSOR		
<u>OBJECT MANAGEMENT</u>		
FETCH		
<u>PROGRAM STRUCTURE</u>		
DO ... END	END_PROC	

RECEPTION SYSTEM OPERATION

RS 400 of computer system network 10 use software called native code modules (to be described below) to enable the user to select options and functions presented on the monitor of personal computer 405, to execute partitioned applications and to process user created events, enabling the partitioned application to interact with interactive system 10. Through this interaction, the user is able to input data into fields provided as part of the display, or may individually select choices causing a standard or personalized page to be built (as explained below) for display on the monitor of personal computer 405. Such inputs will cause RS 400 to interpret events and trigger pre-processors or post-processors, retrieve specified objects, communicate with system components, control user options, cause the display of advertisements on a page, open or close window partitions to provide additional navigation possibilities, and collect and report data about events, including certain types of objects processed. For example, the user may select a particular option, such as opening on closing window partition 275, which is present on the monitor and follow the selection with a completion key stroke, such as ENTER. When the completion key-

stroke is made, the selection is translated into a logical event that triggers the execution of a post-processor, (i.e., a partitioned application program object) to process the contents of the field.

Functions supporting the user-partitioned application interface can be performed using the command bar 290, or its equivalent using pull down windows or an overlapping cascade of windows. These functions can be implemented as part of the RS native functions or can be treated as another partition(s) defined for every page for which an appropriate set of supporting objects exist and remain resident at RS 400. If the functions are part of RS 400, they can be altered or extended by verbs defined in the RS virtual machine that permit the execution of program objects to be triggered when certain functions are called, providing maximum flexibility.

To explain the functions the use of a command bar is assumed. Command bar 290 is shown in FIGS. 3 (a) and 3 (b) and includes a NEXT command 291, a BACK command 292, a PATH command 293, a MENU command 294, and ACTION command 295, a JUMP com-

75

5,347,632

mand 296, a HELP command 297, and an EXIT command 298.

NEXT command 291 causes the next page in the current pageset to be built. If the last page of a pageset has already been reached, NEXT command 291 is disabled by RS 400, avoiding the presentation of an invalid option.

BACK command 292 causes the previous page of the current pageset to be built. If the present page is the first in the pageset, BACK command 292 is disabled, since it is not a valid option.

A filter program can be attached to both the NEXT or BACK functions to modify their implicit sequential nature based upon the value of the occurrence in the object set id.

PATH command 293 causes the next page to be built and displayed from a list of pages that the user has entered, starting from the first entry for every new session.

MENU command 294 causes the page presenting the previous set of choices to be rebuilt.

ACTION command 295 initiates an application dependent operation such as causing a new application partition to be interpreted, a window partition 275 to be opened and enables the user to input any information required which may result in a transaction or selection of another window or page.

JUMP command 296 causes window partition 275 to be opened, allowing the user to input a keyword or to specify one from an index that may be selected for display.

HELP command 297 causes a new application partition to be interpreted such as a HELP window pertaining to where the cursor is positioned to be displayed in order to assist the user regarding the present page, a particular partition, or a field in a page element.

EXIT command 298 causes a LOGOFF page template object PTO to be built, and a page logoff sequence to be presented at RS 400 monitor screen 414.

NAVIGATION INTERFACE

Continuing, as a further feature, the method aspect of the invention includes an improved procedure for searching and retrieving applications from the store of applications distributed throughout network 10; e.g., server 205, cache/concentrator 302 and RS 400. More specifically, the procedure features use of pre-created search tables which represent subsets of the information on the network arranged with reference to the page template objects (PTO) and object-ids of the available applications so that in accordance with the procedure, the relevant tables and associated objects can be provided to and searched at the requesting RS 400 without need to search the entire store of applications on the network. As will be appreciated, this reduces the demand on the server 205 for locating and retrieving applications for display at monitor 412.

In conventional time-sharing networks that support large conventional databases, the host receives user requests for data records; locates them; and transmits them back to the users. Accordingly, the host is obliged to undertake the data processing necessary to isolate and supply the requested information. And, as noted earlier, where large numbers of users are to be served, the many user requests can bottleneck at the host, taxing resources and leading to response slowdown.

Further, users have experienced difficulty in searching data bases maintained on conventional time-sharing networks. For example, difficulties have resulted from

76

the complex and varied way previously known database suppliers have organized and presented their information. Particularly, some database providers require searching be done only in selected fields of the data base, thus requiring the user to be fully familiar with the record structure. Others have organized their databases on hierarchical structures which require the user understand the way the records are grouped. Still further, yet other database suppliers rely upon keyword indices to facilitate searching of their records, thus requiring the user to be knowledgeable regarding the particular keywords used by the database provider.

The method aspect of the present invention, however, serves to avoid such difficulties. In the preferred embodiment, the invention includes procedures for creating preliminary searches which represent subsets of the network applications users are believed likely to investigate. Particularly, in accordance with these procedures, for the active applications available on network 10, a library of tables is prepared, and maintained within each of which a plurality of so called "keywords" are provided that are correlated with page template objects and object-ids of the entry screen (typically the first screen) for the respective application. In the preferred embodiment, approximately 1,000 tables are used, each having approximately 10 to 20 keywords arranged in alphabetical order to abstract the applications on the network. Further, the object-id for each table is associated with a code in the form of a character string mnemonic which is arranged in a set of alphabetically sequenced mnemonics termed the sequence set so that on entry of a character string at an RS 400, the object-id for the relevant keyword table can be obtained from the sequence set. Once the table object-id is identified, the keyword table corresponding to the desired subset of the objects and associated applications can then be obtained from network 10. Subsequently the table can be presented to the user's RS 400, where the RS 400 can provide the data processing required to present the potentially relevant keywords, objects and associated applications to the user for further review and determination as to whether more searching is required. As will be appreciated, this procedure reduces demand on server 205 and thereby permits it to be less complex and costly, and further, reduces the likelihood of host overtaxing that may cause network response slowdown.

As a further feature of this procedure, the library of keywords and their associated PTOs and-objects may be generated by a plurality of operation which appear at the user's screen as different search techniques. This permits the user to select a search technique he is most comfortable with, thus expediting his inquiry.

More particularly, in accordance with the invention, the user is allowed to invoke the procedure by calling up a variety of operation. The various operations have different names and seemingly present different search strategies. Specifically, the user may invoke the procedure by initiating a "Jump" command at RS 400. Thereafter, in connection with the Jump operation, the user, when prompted, may enter a word of the user's choosing at monitor screen 414 relating to the matter he is interested in locating; i.e., a subject matter search of the network applications. Additionally, the users may invoke the procedure by alternatively calling up an operation termed "Index" with selection of the Index command. When selected, the Index command presents the user with an alphabetical listing of keywords from the tables

77

noted above which the user can select from; i.e., an alphabetical search of the network applications. Further, the user may evoke the procedure by initiating an operation termed "Guide." By selecting the Guide command, the user is provided with a series of graphic displays that presents a physical description of the network applications; e.g., department floor plan for a store the user may be electronically shopping in. Still further, the user may invoke the procedures by initiating an operation termed "Directory." By selecting the Directory command, the user is presented with the applications available on the network as a series of hierarchical menus which present the content of the network information in commonly understood categories. Finally, the user may invoke the procedure by selecting the "Path" command, which accesses a list of keywords the user has previously selected; i.e., a personally tailored form of the Index command described above. As described hereafter, Path further includes a Viewpath operation which permits the user to visually access and manage the Path list of keywords. In preferred form, where the user has not selected a list of personalized keywords, a default set is provided which includes a predetermined list and associated applications deemed by network 10 as likely to be of interest to the user.

In accordance with the invention, this ability to convert these apparently different search strategies in a single procedure for accessing pre-created library tables is accomplished by translating the procedural elements of the different search techniques into a single set of procedures that will produce a mnemonic which can first be searched at the sequence set, described above to identify the object-id for the appropriate library table and, thereafter, enable access of the appropriate table to permit selection of the desired keyword and associated PTO and object-ids. Thus, while the search techniques may appear different to the user, and in fact accommodate the user's preferences and sophistication level, they nonetheless invoke the same efficient procedure of relying upon pre-created searches which identify related application PTOs and object ids so that the table and objects may be collected and presented at the user's RS 400 where they can be processed, thereby relieving server 205.

In preferred form, however, in order to enhance presentation speed the Guide operation is specially configured. Rather than relating the keyword mnemonic to a sequence set to identify the table object-id and range of keywords corresponding to the entry PTO and associated object-ids, the Guide operation presents a series of overlapping windows that physically describe the "store" in which shopping is being conducted or the "building" from which information is being provided. The successive windows increase in degree of detail, with the final window presenting a listing of relevant keywords. Further, the PTO and object-ids for the application entry screen are directly related to the graphic presentation of the keywords. This eliminates the need to provide variable fields in the windows for each of the keywords and enables the entry screen to be correlated directly with the window graphic. As will be appreciated, this reduces the number of objects that would otherwise be required to be staged at RS 400 to support pretention of the keyword listing at monitor screen 414, and thus speeds network response.

A more detailed understanding of the procedure may be had upon a reading of the following description and review of accompanying FIGS. 2a, 3a and particularly

5,347,632

78

FIG. 11 which presents a flow diagram of the search procedure.

To select a particular partitioned application from among thousands of such applications residing either at the RS 400 or within delivery system 20, the present invention avoids the need for a user to know or understand, prior to a search, the organization of such partitioned applications and the query techniques necessary to access them. This is accomplished using a collection of related commands, as described below.

The Jump command 296 as been in FIG. 3a, can be selected, by the user from command bar 290. When Jump command 296 is selected, a window partition 275 is opened. In window 275, the user is presented and may select from a variety of displayed options that include among others, the Directory command, the Index command, and the Guide command, which when selected, have the effect noted above. Additionally, the user can select a command termed Viewpath which will presents the keywords that currently make up the list of keywords associated with the user's Path command, and from which list the user can select a desired keyword. Alternatively, the user may also enter a keyword at display field 270 within window partition 275 as a "best guess" of the mnemonic character string that is assigned to a partitioned application the user desires (e.g., the user may input such english words as "news," "pet food," "games," etcetera). Where the user enters a character string it is displayed in field 270, and then searched by RS 400 native code (discussed below) against the sequence sets above noted to identify the object-id for the appropriate table of keywords (not shown) that RS 400 may request from host 205. While as note above, a table may include 10 to 20 keywords, in the preferred embodiment, for the sake of speed and convenience, a the typical keyword table includes approximately 12 keywords.

If the string entered by the user matches a keyword existing on one of the keyword tables, and is thus associated with a specific PTO, RS 400 fetches and displays associated objects of the partitioned applications and builds the entry page in accordance with the page composition dictated by the target PTO.

If the string entered by the user does not match a specific keyword, RS 400 presents the user with the option of displaying the table of keywords approximating the specific keyword. The approximate keywords are presented as initialized, cursorable selector fields of the type provided in connection with a Index command. The user may then move the cursor to the nearest approximation of the mnemonic he originally selected, and trigger navigation to the PTO associated with that keyword, navigation being as described hereafter in connection with the RS native code.

If, instead of selecting the Jump command, the user selects the Index command, RS 400 will retrieve the keyword table residing at RS 400, and will again build a page with initialized, cursorable fields of keywords. The table fetched upon invoking the Index command will be comprised of alphabetic keywords that occur within the range of the keywords associated with the page template object (PTO) from which the user invoked the Index con, hand. As discussed above, the user may select to navigate to any of this range of PTOs by selecting the relevant keyword from the display. Alternatively, the user can, thereafter, select another range of alphabetical keywords by entering an appropriate character string in a screen field provided or move forward

or backward in the collection by selecting corresponding option.

By selecting the Directory command, RS 400 can be caused to fetch a table of keywords, grouped by categories, to which the PTO of the current partitioned application (as specified by the object set field 630 of the current PEO) belongs. Particularly, by selecting the Directory command, RS 400, is causes to displays a series of screens each of which contains alphabetically arranged general subject categories from which the user may select. Following selection of a category, a series of keywords associated with the specified category are displayed in further screens together with descriptive statements about the application associated with the keywords. Thereafter, the user can, in the manner previously discussed with regard to the Index command, select from and navigate to the PTOs of keywords which are related to the present pageset by subject.

The Guide command provides a navigation method related to a hierarchical organization of applications provided on network 10, and are described by a series of sequentially presented overlaying windows of a type known in the art, each of which presents an increasing degree of detail for a particular subject area, terminating in a final window that gives keywords associated with the relevant applications. The Guide command makes use of the keyword segment which describes the location of the PTO in a hierarchy (referred to, in the preferred embodiment, as the "BFD," or Building-Floor-Department) as well as an associated keyword character string. The BFD describes the set of menus that are to be displayed on the screen as the sequence of pop-up windows. The Guide command may be invoked by requesting it from the Jump window described above, or by selecting the Menu command on Command Bar 290. As noted above, in the case of the Guide command, the PTO and object-ids for the application entry screen are directly associated with the graphic of the keyword presented in the final pop-up window. This enables direct access of the application entry screen without need to access the sequence set and keyword table, and thus, reduces response time by reducing the number of objects that must be processed at RS 400.

Activation of the Path command accesses the user's list of pre-selected keywords without their display, and permits the user to step through the list viewing the respective applications by repeatedly invoking the Path command. As will be appreciated, the user can set a priority for selecting keywords and viewing their associated applications by virtue of where on the list the user places the keywords. More specifically, if the user has several application of particular interest; e.g., news, weather, etc., the user can place them at the top of the list, and quickly step through them with the Path command. Alternatively, the user can view and randomly access the keywords of his list with the Viewpath operation noted above. On activation of Viewpath, the user's Path keywords are displayed and the user can cursor through them in a conventional manner to select a desired one. Further, the user can amend the list as desired by changing the keywords on the list and/or adjusting their relative position. This is readily accomplished by entering the amendments to the list presented at the screen 414 with a series of amendment options presented in a conventional fashion with the list. As noted, the list may be personally selected by the user in

the manner described, or created as a default by network 10.

Collectively, the Jump command, Index command, Directory command, Guide command, and Path command as described enable the user to quickly and easily ascertain the "location" of either the partitioned application presently displayed or the "location" of a desired partitioned application. "Location," as used in reference to the preferred embodiment of the invention, means the specific relationships that a particular partitioned application bears to other such applications, and the method for selecting particular partitioned applications from such relationships. The techniques for querying a database of objects, embodied in the present invention, is an advance over the prior art, insofar as no foreknowledge of either database structure or query technique or syntax is necessary, the structure and search techniques being made manifest to the user in the course of use of the commands.

RS APPLICATION PROTOCOL

RS protocol defines the way the RS supports user application conversation (input and output) and the way RS 400 processes a partitioned application. Partitioned applications are constructed knowing that this protocol will be supported unless modified by the application. The protocol is illustrated FIG. 6. The boxes in FIG. 6 identify processing states that the RS 400 passes through and the arrows indicate the transitions permitted between the various states and are annotated with the reason for the transition.

The various states are: (A) Initialize RS, (B) Process Objects, (C) Interpretively Execute Pre-processors, (D) Wait for Event, (E) Process Event, and (F) Interpretively Execute Function Extension and/or Post-processors.

The transitions between states are: (1a) Logon Page Template Object identification (PTO-id), (1b) Object-id, (2) Trigger Program Object identification (PO-id) & return, (3) Page Partition Template (PPT) or Window Stack Processing complete, (4) Event occurrence, and (5) Trigger PDO-id and return.

Transition (1a) from Initialize RS (A) to Process Objects (B) occurs when an initialization routine passes the object-id of the logon PTO to object interpreter 435, when the service is first invoked. Transition (1b) from Process Event (E) to Process Objects (B) occurs whenever a navigation class event causes a new PTO object-id to be passed to object interpreter 435; or when a open window event (verb or function key) occurs passing a window Object-id to the object interpreter 435; or a close window event (verb or function key) occurs causing the current top-most window to be closed.

While in the process object state, object interpreter 435 will request any objects that are identified by external references in call segments. Objects are processed by parsing and interpreting the object and its segments according to the specific object architecture. As object interpreter 435 processes objects, it builds a linked list structure called a Page Processing Table (PPT), shown in FIG. 10, to reflect the structure of the page, each page partition, Page Elements Objects (PEOs) required, Program Objects (POs) required and each window object (WO) that could be called. Object interpreter 435 requests all objects required to build a page except objects that could be called as the result of some event, such as a HELP window object.

81

Transition (2) from Process Objects (B) to Interpretively Execute Pre-processors (C) occurs when the object interpreter 435 determines that a pre-processor is to be triggered. Object processor 436 then passes the object-id of the Program Object (PO) to the TBOL interpreter 438. TBOL interpreter 438 uses the RS virtual machine to interpretively execute the Program Object (PO). The PO can represent either a selector or an initializer. When execution is complete, a transition automatically occurs back to Process Objects (B).

Selectors are used to dynamically link and load other objects such as Page Element Object (PEO) or other POs based upon parameters that they are passed when they are called. Such parameters are specified in call segments or selector segments. This feature enables RS 400 to conditionally deliver information to the user base upon predetermined parameters, such as his personal demographics or locale. For example, the parameters specified may be the transaction codes required to retrieve the user's age, sex, and personal interest codes from records contained in user profiles stored at the switch/file server layer 200.

Initializers are used to set up the application processing environment for a partitioned application and determine what events RS 400 may respond to and what the action will be.

Transition (3) from Process Objects (B) to Wait for Event (D) occurs when object interpreter 435 is finished processing objects associated with the page currently being built or opening or closing a window on a page. In the Wait for Event state (D), input manager, which in the preferred form shown is comprised of keyboard manager 434 seen in FIG. 8, accepts user inputs. All keystrokes are mapped from their physical codes to logical keystrokes by the Keyboard Manager 434, representing keystrokes recognized by the RS virtual machine.

When the cursor is located in a field of a page element, keystrokes are mapped to the field and the Partitioned External Variable (PEV) specified in the Page Element Object (PEO) field definition segment by the cooperative action of Keyboard manager 434 and display manager 461. Certain inputs, such as RETURN or mouse clicks in particular fields, are mapped to logical events by keyboard manager 434, which are called completion (or commit) events. Completion events signify the completion of some selection or specification process associated with the partitioned application and trigger a partition level and/or page level post-processor to process the "action" parameters associated with the user's selection and commit event.

Such parameters are associated with each possible choice or input, and are set up by the earlier interpretive execution of an initializer pre-processor in state (C). Parameters usually specify actions to perform a calculation such as the balance due on an order of several items with various prices using sales tax for the user's location, navigate to PTO-id, open window WO-id or close window. Actions parameters that involve the specification of a page or window object will result in transition (1b) to the Process Objects (B) state after the post-processor is invoked as explained below.

Function keys are used to specify one or more functions which are called when the user strikes these keys. Function keys can include the occurrence of logical events, as explained above. Additionally, certain functions may be "filtered", that is, extended or altered by SET_FUNCTION or TRIGGER_FUNCTION

5,347,632

82

verbs recognized by the RS virtual machine. Function keys cause the PO specified as a parameter of the verb to be interpretively executed whenever that function is called. Applications use this technique to modify or extend the functions provided by the RS.

Transition (5) from Process Event (E) to Interpretively Execute Pre-processors (F) occurs when Process Event State determines that a post-processor or function extension PDO is to be triggered. The object id of the Program Object PO is then passed to the TBOL interpreter 438. The TBOL interpreter 438 uses the RS virtual machine to interpretively execute the PO. When execution is complete a transition automatically occurs back to Process Event (E).

RECEPTION SYSTEM SOFTWARE

The reception system 400 software is the interface between the user of personal computer 405 and interactive network 10. The object of reception system software is to minimize mainframe processing, minimize transmission across the network, and support application extensibility and portability.

RS 400 software is composed of several layers, as shown in FIG. 7. It includes external software 451, which is composed of elements well known to the art such as device drivers, the native operating systems; i.e., MS-DOS, machine-specific assembler functions (in the preferred embodiment; e.g., CRC error checking), and "C" runtime library functions; native software 420; and partitioned applications 410.

Again with reference to FIG. 7, native software 420 is compiled from the "C" language into a target machine-specific executable, and is composed of two components: the service software 430 and the operating environment 450. Operating environment 450 is comprised of the Logical Operating System 432, or LOS; and a multitasker 433. Service software 430 provides functions specific to providing interaction between the user and interactive network 10, while the operating environment 450 provides pseudo multitasking and access to local physical resources in support of service software 430. Both layers of native software 420 contain kernel, or device independent functions 430 and 432, and machine-specific or device dependent functions 433. All device dependencies are in code resident at RS 400, and are limited to implementing only those functions that are not common across machine types, to enable interactive network 10 to provide a single data stream to all makes of personal computer which are of the IBM or IBM compatible type. Source code for the native software is given below, and provides a detailed description of the software features for the reception system.

Service software 430 is comprised of modules, which are device-independent software components that together obtain, interpret and store partitioned applications existing as a collection of objects. The functions performed by, and the relationship between, the service software 430 module is shown in FIG. 8 and discussed further below.

Through facilities provided by LOS 432 and multitasker 433, here called collectively operating environment 450, provides device-independent multitasking and access to local machine resources, such as multitasking, timers, buffer management, dynamic memory management, file storage and access, keyboard and mouse input, and printer output. The operating environment 450 manages communication and synchronization of service software 430, by supporting a request/re-

5,347,632

83

sponse protocol and managing the interface between the native software 420 and external software 437.

Applications software layer 410 consists of programs and data written in an interpretive language, "TRIN-TEX Basic Object Language" or "TBOL," described above, which was written specifically for use in RS 400 and interactive network 10 to facilitate videotext-specific commands and achieve machine-independent compiling. TBOL is constructed as objects, which in interaction with one another comprise partitioned applications.

RS native software provides a virtual machine interface for partitioned applications, such that all objects comprising partitioned applications "see" the same machine. RS native software provides support for the following functions: (1) keyboard and mouse input; (2) text and graphics display; (3) application interpretation; (4) application database management; (5) local application storage; (6) network and link level communications; (7) user activity data collection; and (8) advertisement management.

With reference to FIG. 8, service software 430 is comprised of the following modules: start-up (not shown); keyboard manger 434; object interpreter 435; TBOL interpreter 438; object storage facility 439; display manager 461; data collection manager 441; ad manager 442; object/communications manager interface 443; link communications manager 444; and fatal error manager 469. Each of these modules has responsibility for managing a different aspect of RS 400.

Startup reads RS 400 customization options into RAM, including modem, device driver and telephone number options, from the file CONFIG.SM. Startup invokes all RS 400 component startup functions, including navigation to the first page, a logon screen display containing fields initialized to accept the user's id and password. Since Startup is invoked only at initialization, for simplicity, it has not been shown in FIG. 8.

The principal function of keyboard manger 434 is to translate personal computer dependent physical input into a consistent set of logical keys and to invoke processors associated with these keys. Depending on the LOS key, and the associated function attached to it, navigation, opening of windows, and initiation of filter or post-processor TBOL programs may occur as the result input events handled by the keyboard manger 434. In addition, keyboard manger 434 determines inter and intra field cursor movement, and coordinates the display of field text and cursor entered by the user with display manager 461, and sends information regarding such inputs to data collection manager 441.

Object interpreter 435 is responsible for building and recursively processing a table called the "Page Processing Table," or PPT. Object interpreter 435 also manages the opening and closing of windows at the current page. Object interpreter 435 is implemented as two sub-components: the object processor 436 and object scanner 437.

Object processor 436 provides an interface to keyboard manger 434 for navigation to new pages, and for opening and closing windows in the current page. Object processor 436 makes a request to Object storage facility 439 for a page template object (PTO) or window object (WO), as requested by keyboard manger 434, and for objects and their segments which comprise the PTO or WO returned by Object storage facility 439 to object processor 436. Based on the particular segments comprising the object(s) making up the new PTO

84

or WO, object processor 436 builds or adds to the PPT, which is an internal, linkedlist, global data structure reflecting the structure of the page or Page Format Object (PFO), each page partition or Page Element Object (PEO), and Program Objects (POs) required and each window object that could be called. Objects are processed by parsing and interpreting each object and its segment(s) according to their particular structure as formalized in the data object architecture (DOA). While in the process object state, object processor 436 will request any objects specified by the PTO that are identified by external references in call segments (e.g. field level program call 518, page element selector call 524, page format call 526 program call 532, page element call 522 segments) of such objects, and will, through a request to TBOL interpreter 438, fire initializers and selectors contained in program data segments of all PTO constituent program objects, at the page, element, and field levels. Object processor 436 requests all objects required to build a page, except objects that could only be called as the result of some event external to the current partitioned application, such as a HELP window object. When in the course of building or adding to the PPT and opening/closing WO, object processor encounters a call to an object with object id "ADSL0T," it fetches the next advertisement object 510 from ad manager 442, and sends to display manager 461 for display to the user presentation data segments 530 contained in the objects constituent of the PTO, WO and advertisement object. Object processor also passes to data collection manager 441 all object-ids that were requested and object ids that were viewed. Upon completion of page or window processing, object processor 436 enters the wait for event state, and control is returned to keyboard manger 434.

The second component of object interpreter 435, object scanner 437, provides a file-like interface, shared with object storage facility 439, to objects currently in use at RS 400, to enable object processor 436 to maintain and update the PPT. Through facilities provided by object scanner 437, object processor recursively constructs a page or window in the requested or current partitioned application, respectively.

Object storage facility 439 provides an interface through which object interpreter 435 and TBOL interpreter 438 either synchronously request (using the TBOL verb operator "GET") objects without which processing in either module cannot continue, or asynchronously request (using the TBOL verb operator "FETCH") objects in anticipation of later use. Object storage facility 439 returns the requested objects to the requesting module once retrieved from either local store 440 or interactive network 10. Through control structures shared with the object scanner 437, object storage facility determines whether the requested object resides locally, and if not, makes an attempt to obtain it from interactive network 10 through interaction with link communications manager 444 via object-/communications manager interface 443.

When objects are requested from object storage facility 439, only the latest version of the object will be provided to guarantee currency of information to the user. Object storage facility 439 assures currency by requesting version verification from network 10 for those objects which are available locally and by requesting objects which are not locally available from delivery system 20 where currency is maintained.

5,347,632

85

Version verification increases response time. Therefore, not all objects locally available are version checked each time they are requested. Typically, objects are checked only the first time they are requested during a user session. However, there are occasions, as for example in the case of objects relating to news applications, where currency is always checked to assure integrity of the information.

The frequency with which the currency of objects is checked depends on factors such as the frequency of updating of the objects. For example, objects that are designated as ultrastable in a storage control parameter in the header of the object are never version checked unless a special version control object sent to the RS as part of logon indicates that all such objects must be version checked. Object storage facility 439 marks all object entries with such a stability category in all directories indicating that they must be version checked the next time they are requested.

Object storage facility 439 manages objects locally in local store 440, comprised of a cache (segmented between available RAM and a fixed size disk file), and stage (fixed size disk file). Ram and disk cached objects are retained only during user sessions, while objects stored in the stage file are retained between sessions. The storage control field, located in the header portion of an object, described more fully hereafter as the object "storage candidacy", indicates whether the object is stageable, cacheable or trashable.

Stageable objects must not be subject to frequent change or update. They are retained between user sessions on the system, provided storage space is available and the object has not discarded by a least-recently-used (LRU) algorithm of a conventional type; e.g., see *Operating System Theory*, by Coffman, Jr. and Denning, Prentice Hall Publishers, New York, 1973, which in, accordance with the invention, operates in combination with the storage candidacy value to determine the object storage priority, thus rendering the stage self-configuring as described more fully hereafter. Over time, the self-configuring stage will have the effect of retaining within local disk storage those objects which the user has accessed most often. The objects retained locally are thus optimized to each individual user's usage of the applications in the system. Response time to such objects is optimized since they need not be retrieved from the interactive computer system.

Cacheable objects can be retained during the current user session, but cannot be retained between sessions. These objects usually have a moderate update frequency. Object storage facility 439 retains objects in the cache according to the LRU storage retention algorithm. Object storage facility 439 uses the LRU algorithm to ensure that objects that are least frequently used forfeit their storage to objects that are more frequently used.

Trashable objects can be retained only while the user is in the context of the partitioned application in which the object was requested. Trashable objects usually have a very high update frequency and must not be retained to ensure that the user has access to the most current data.

More particularly and, as noted above, in order to render a public informational and transactional network of the type considered here attractive, the network must be both economical to use and fast. That is to say, the network must supply information and transactional support to the user at minimal costs and with a minimal

86

response time. In accordance with the present invention, these objectives are sought to be achieved by locating as many information and transactional support objects which the user is likely to request, as close to the user as possible; i.e., primarily at the user's RS 400 and secondarily at delivery system 20. In this way, the user will be able to access objects required to support a desired application with minimal intervention of delivery system 20, thus reducing the cost of the session and speeding the response time.

However, the number of objects that can be maintained at RS 400 is restricted by at least two factors: the RS 400 storage capacity; i.e., RAM and disk sizes, and the need to maintain the stored objects current.

In accordance with the method aspect of the invention, in order to optimize the effectiveness of the limited storage space at RS 400, the collection of objects is restricted to those likely to be requested by the user; i.e., tailored to the user's tastes—and to those least likely to be time sensitive; i.e., objects which are stable. To accomplish this, objects are coded for storage candidacy to identify when they will be permitted at RS 400, and subject to the LRU algorithm to maintain presence at RS 400. Additionally, to assure currency of the information and transaction support provided at RS 400, objects are further coded for version identification and checking in accordance with a system of priorities that are reflected in the storage candidacy coding.

Specifically, to effect object storage management, objects are provided with a coded version id made up of the storage control byte and version control bytes identified above as elements of the object header, specifically, bytes 16 and 18 shown in FIG. 4b. In preferred form, the version id is comprised of bytes 16 and 18 to define two fields, a first 13 bit field to identify the object version and a second three bite field to identify the object storage candidacy.

In this arrangement, the storage candidacy value of the object is addressed to not only the question of storage preference but also object currency. Specifically, the storage candidacy value establishes the basis upon which the object will be maintained at RS 400 and also identifies the susceptibility of the object to becoming stale by dictating when the object will be version checked to determine currency.

The version value of the object on the other hand, provides a parameter that can be checked against predetermined values available from delivery system 20 to determine whether an object stored at RS 400 is sufficiently current to permit its continued use, or whether the object has become stale and needs to be replaced with a current object from delivery system 20.

Still further, in accordance with the invention, object storage management procedure further includes use of the LRU algorithm, for combination with the storage and version coding to enable discarding of objects which are not sufficiently used to warrant retention, thus personalizing the store of objects at RS 400 to the user's tastes. Particularly, object storage facility 439, in accordance with the LRU algorithm maintains a usage list for objects. As objects are called to support the user's applications requests, the objects are moved to the top of a usage list. As other objects are called, they push previously called objects down in the list. If an object is pushed to the bottom of the list before being recalled, it will be forfeited from the list if necessary to make room for the next called object. As will be appreciated, should a previously called object be again called

5,347,632

87

before it is displaced from the list, it will be promoted to the top of the list, and once more be subject to depression in the list and possible forfeiture as other objects are called.

As pointed out above, in the course of building the screens presented to the user, objects will reside at various locations in RS 400. For example, objects may reside in the RS 400 RAM where the object is supporting a particular application screen then running or in a cache maintained at either RAM or disk 424 where the object is being held for an executing application or staged on the fixed size file on disk 424 noted above where the object is being held for use in application likely to be called by the user in the future.

In operation, the LRU algorithm is applied to all these regions and serves to move an object from RAM to disk cache to disk file, and potentially off RS 400 depending on object usage.

With regard to the storage candidacy value, in this arrangement, the objects stored at RS 400 include a limited set of permanent objects; e.g., those supporting logon and logoff, and other non-permanent objects which are subject to the LRU algorithm to determine whether the objects should be forfeited from RS 400 as other objects are added. Thus, in time, and based on the operation of the LRU algorithm and the storage candidacy value, the collection of objects at RS 400 will be tailored to the usage characteristics of the subscriber; i.e., self-configuring.

More particularly, the 3-bit field of the version id that contains the storage candidacy parameter can have 8 different values. A first candidacy value is applied where the object is very sensitive to time; e.g., news items, volatile pricing information such as might apply to stock quotes, etc. In accordance with this first value, the object will not be permitted to be stored On RS 400, and RS 400 will have to request such objects from delivery system 20 each time it is accessed, thus, assuring currency. A second value is applied where the object is sensitive to time but less so than the first case; e.g., the price of apples in a grocery shopping application. Here, while the price might change from day to day, it is unlikely to change during a session. Accordingly the object will be permitted to persist in RAM or at the disk cache during a session, but will not be permitted to be maintained at RS 400 between sessions.

Continuing down the hierarchy of time sensitivity, where the object concerns information sufficiently stable to be maintained between sessions, a third storage candidacy value is set to permit the object to be stored at RS 400 between sessions, on condition that the object will be version check the first time it is accessed in a subsequent session. As will be appreciated, during a session, and under the effect of the LRU algorithm, lack of use at RS 400 of the object may result in it being forfeited entirely to accommodate new objects called for execution at RS 400.

Still further, a fourth value of storage candidacy is applied where the object is considered sufficiently stable as not to require version checking between sessions; e.g., objects concerning page layouts not anticipated to change. In this case, the storage candidacy value may be encoded to permit the object to be retained from session to session without version checking. Here again, however, the LRU algorithm may cause the object to forfeit its storage for lack of use.

Where the object is of a type required to be stored at RS 400, as for example, objects needed to support stan-

88

dard screens, it is coded for storage between sessions and not subject to the LRU algorithm forfeiture. However, where such objects are likely to change in the future they may be required to be version checked the first time they are accessed in a session and thus be given a fifth storage candidacy value. If, on the other hand, the required stored object is considered likely to be stable and not require even version checking; e.g., logon screens, it will be coded with a sixth storage candidacy value for storage without version checking so as to create a substantially permanent object.

Continuing, where a RS 400 includes a large amount of combined RAM and disk capacity, it would permit more objects to be stored. However, if objects were simply coded in anticipation of the larger capacity, the objects would potentially experience difficulty, as for example, undesired forfeiture due to capacity limitations if such objects were supplied to RS 400 units having smaller RAM and disk sizes. Accordingly, to take advantage of the increased capacity of certain RS 400 units without creating difficulty in lower capacity units, objects suitable for storage in large capacity units can be so coded for retention between sessions with a seventh and eighth storage candidacy value depending upon whether the stored large capacity object requires version checking or not. Here, however, the coding will be interpreted by smaller capacity units to permit only cacheable storage to avoid undesirable forfeiture that might result from over filling the smaller capacity units.

Where an object is coded for no version checking need may nonetheless arise for a version check at some point. To permit version checking of such objects, a control object is provided at RS 400 that may be version checked on receipt of a special communication from delivery system 20. If the control object fails version check, then a one shot version checking attribute is associated with all existing objects in RS 400 that have no version checking attributes. Thereafter, the respective objects are version checked, the one shot check attribute is removed and the object is caused to either revert to its previous state if considered current or be replaced if stale.

Still further, objects required to be stored at RS 400 which are not version checked either because of lack of requirement or because of no version check without a control object, as described above, can accumulate in RS 400 as dead objects. To eliminate such accumulation, all object having required storage are version checked over time. Particularly, the least recently used required object is version checked during a session thus promoting the object to the top of the usage list if it is still to be retained at RS 400. Accordingly, one such object will be checked per session and over time, all required objects will be version checked thereby eliminating the accumulation of dead objects.

However, in order to work efficiently, the version check attribute of the object should be ignored, so that even required object can be version checked. Yet, in certain circumstances, e.g., during deployment of new versions of the reception system software containing new objects not yet supported on delivery system 20 which may be transferred to the fixed storage file of RS 400 when the new version is loaded, unconditional version checking may prematurely deletes the object from the RS 400 as not found on delivery system 20. To avoid this problem, a sweeper control segment in the control object noted above can be used to act as a switch to turn the sweep of dead objects on and off.

With respect to version checking for currency, where an object stored at RS 400 is initially fetched or accessed during a session, a request to delivery system 20 is made for the object by specifying the version id of the object stored at RS 400.

In response, delivery system 20 will advise the reception system 400 either that the version id of the stored object matches the currency value; i.e., the stored object is acceptable, or deliver a current object that will replace the stored object shown to be stale. Alternatively, the response may be that the object was not found. If the version of the stored object is current, the stored object will be used until verified again in accordance with its storage candidacy. If the stored object is stale, the new object delivered will replace the old one and support the desired screen. If the response is object not found, the stored object will be deleted.

Therefore, based on the above description, the method aspect of the invention is seen to include steps for execution at storage facility 439 which enables object reception, update and deletion by means of a combination of operation of the LRU algorithm and interpretation of the storage candidacy and version control values. In turn, these procedures cooperate to assure a competent supply of objects at RS 400 so as to reduce the need for intervention of delivery system 20, thus reducing cost of information supply and transactional support so as to speed the response to user requests.

TBOL interpreter 438 provides the means for executing program objects, which have been written using an interpretive language, TBOL described above. TBOL interpreter 438 interprets operators and operand contained in program object 508, manages TBOL variables and data, maintains buffer and stack facilities, and provides a runtime library of TBOL verbs.

TBOL verbs provide support for data processing, program flow control, file management, object management, communications, text display, command bar control, open/close window, page navigation and sound. TBOL interpreter also interacts with other native modules through commands contained in TBOL verbs. For example: the verb "navigate" will cause TBOL interpreter 438 to request object interpreter 435 to build a PPT based on the PTO id contained in the operand of the NAVIGATE verb; "fetch" or "GET" will cause TBOL interpreter 438 to request an object from object storage facility 439; "SET_FUNCTION" will assign a filter to events occurring at the keyboard manger 434; and "FORMAT," "SEND," and "RECEIVE" will cause TBOL interpreter 438 to send application level requests to object/communications manager interface 433.

Data areas managed by TBOL interpreter 438 and available to TBOL programs are Global External Variables (GEVs), Partition External Variables (PEVs), and Runtime Data Arrays (RDAs).

GEVs contain global and system data, and are accessible to all program objects as they are executed. GEVs provide a means by which program objects may communicate with other program objects or with the RS native code, if declared in the program object. GEVs are character string variables that take the size of the variables they contain. GEVs may preferably contain a maximum of 32,000 variables and are typically used to store such information as program return code, system date and time, or user sex or age. TBOL interpreter 438 stores such information in GEVs when requested by the program which initiated a transaction to obtain these

records from the RS or user's profile stored in the interactive system.

Partition external variables (PEVs) have a scope restricted to the page partition on which they are defined. PEVs are used to hold screen field data such that when PEOs and window objects are defined, the fields in the page partitions with which these objects are to be associated are each assigned to a PEV. When applications are executed, TBOL interpreter 438 transfers data between screen fields and their associated PEV. When the contents of a PEV are modified by user action or by program direction, TBOL interpreter 428 makes a request to display manager 461 to update the screen field to reflect the change. PEVs are also used to hold partition specific application data, such as tables of information needed by a program to process an expected screen input.

Because the scope of PEVs is restricted to program objects associated with the page partition in which they are defined, data that is to be shared between page partitions or is to be available to a page-level processor must be placed in GEVs or RDAs.

RDAs are internal stack and save buffers used as general program work areas. RDAs are dynamically defined at program object "runtime" and are used for communication and transfer of data between programs when the data to be passed is not amenable to the other techniques available. Both GEVs and RDAs include, in the preferred embodiment, 8 integer registers and 8 decimal registers. Preferably, there are also 9 parameter registers limited in scope to the current procedure of a program object.

All variables may be specified as operand of verbs used by the virtual machine. The integer and decimal registers may be specified as operand for traditional data processing. The parameter registers are used for passing parameters to "called" procedures. The contents of these registers are saved on an internal program stack when a procedure is called, and are restored when control returns to the "calling" procedure from the "called" procedure.

TBOL interpreter 438, keyboard manger 434, object interpreter 435, and object storage facility 439, together with device control provided by operating environment 450, have principal responsibility for the management and execution of partitioned applications at the RS 400. The remaining native code modules function in support and ancillary roles to provide RS 400 with the ability display partitioned applications to the user (display manager 461), display advertisements (ad manager 442), to collect usage data for distribution to interactive network 10 for purposes of targeting such advertisements (data collection manager 441), and prepare for sending, and send, objects and messages to interactive network 10 (object/communications manager interface 443 and link communications manager 444) Finally, the fatal error manager exists for one purpose: to inform the user of RS 400 and transmit to interactive network 10 the inability of RS 400 to recover from a system error.

Display manager 461 interfaces with a decoder using the North American Presentation Level Protocol Syntax (NAPLPS), a standard for encoding graphics data, or text code, such as ASCII, which are displayed on monitor 412 of the user's personal computer 405 as pictorial codes. Codes for other presentation media, such as audio, can be specified by using the appropriate type code in the presentation data segments. Display manager 461 supports the following functions: send

91

5,347,632

92

NAPLPS strings to the decoder; echo text from a PEV; move the cursor within and between fields; destructive or non-destructive input field character deletion; "ghost" and "unghost" fields (a ghosted field is considered unavailable, unghosted available); turn off or on the current field cursor; open, close, save and restore bit maps for a graphics window; update all current screen fields by displaying the contents of their PEVs, reset the NAPLPS decoder to a known state; and erase an area of the screen by generating and sending NAPLPS to draw a rectangle over that area. Display manager 461 also provides a function to generate a beep through an interface with a machine-dependent sound driver.

Ad manager 442 is invoked by object interpreter 435 to return the object-id of the next of the next available advertisement to be displayed. Ad manager 442 maintains a queue of advertisement object id's targeted to the specific user currently accessing interactive network 10. Advertisement objects are pre-fetched from interactive system 10 from a personalized queue of advertisements that is constructed using data previously collected from user generated events and/or reports of objects used in the building of pages or windows, compiled by data collection manager 466 and transmitted to interactive system 10.

Advertisement objects 510 are PEOs that, through user invocation of a "LOOK" command, cause navigation to partitioned applications that may themselves support, for example, ordering and purchasing of merchandise.

An advertisement list, or "ad queue," is requested in a transaction message to delivery system 20 by ad manager 442 immediately after the initial logon response. The logon application at RS 400 places the advertisement list in a specific RS global storage area called a SYS_GEV (system global external variable), which is accessible to all applications as well as to the native RS code). The Logon application also passes the first two ad object id's to object storage facility 439 to be requested. At logon, no advertisement objects will be available RS local storage facilities 440, so they must be requested from interactive network 10.

In a preferred embodiment, the following parametric values are established for ad manager 442: advertisement queue capacity, replenishment threshold for advertisement object id's and replenishment threshold for number of outstanding pre-fetched advertisement objects. These parameters are set up in GEVs of the RS virtual machine by the logon application program object from the logon response from high function system 110. The parameters are then also accessible to the ad manager 442. Preferred values are an advertisement queue capacity of 15, replenishment value of 10 empty queue positions and a prefetched advertisement threshold of 3.

Ad manager 442 pre-fetches advertisement object by passing advertisement object id's from the advertisement queue to object storage facility 439 which then retrieves the object from the interactive system if the object is not available locally. Advertisements are pre-fetched, so they are available in RS local store 440 when requested by object id by object interpreter 435 while it is building a page. The Ad manager 443 pre-fetches additional advertisement objects whenever the number of pre-fetched advertisements, not used by object interpreter 435 falls below the pre-fetch advertisement threshold.

Whenever the advertisement queue has more empty positions than replenishment threshold, a transaction is made to the advertisement queue application in high function system 110 shown in FIG. 2, via object/communications manager interface 443 for a number of advertisement object id's equal to the threshold. A response message includes a list of advertisement object id's, which ad manager 442 enqueues.

Object interpreter 435 requests the object id of the next advertisement from ad manager 442 when object interpreter 435 is building a page and encounters an object call for a partition and the specified object-id equals the code word, "ADSLOT." If this is the first request for an advertisement object id that ad manager 442 has received during this user's session, ad manager 442 moves the advertisement list from the GEV into its own storage area, which it uses as an advertisement queue and sets up its queue management pointers, knowing that the first two advertisement objects have been pre-fetched.

Ad manager 442 then queries object storage facility 439, irrespective of whether it was the first request of the session. The query asks if the specified advertisement object id pre-fetch has been completed, i.e., is the object available locally at the RS. If the object is available locally, the object-id is passed to object interpreter 435, which requests it from object storage facility 439. If the advertisement object is not available in local store 440, ad manager 442 attempts to recover by asking about the next ad that was pre-fetched. This is accomplished by swapping the top and second entry in the advertisement queue and making a query to object storage facility 439 about the new top advertisement object id. If that object is not yet available, the top position is swapped with the third position and a query is made about the new top position.

Besides its ability to provide advertisements that have been targeted to each individual user, two very important response time problems have been solved by ad manager 442 of the present invention. The first is to eliminate from the new page response time the time it takes to retrieve an advertisement object from the host system. This is accomplished by using the aforementioned pre-fetching mechanism.

The second problem is caused by pre-fetching, which results in asynchronous concurrent activities involving the retrieval of objects from interactive system 10. If an advertisement is prefetched at the same time as other objects required for a page requested, the transmission of the advertisement object packets could delay the transmission of the other objects required to complete the current page by the amount of time required to transmit the advertisement object(s). This problem is solved by the structuring the requests from object interpreter 435 to the ad manager 442 in the following way:

1. Return next object-id of pre-fetched advertisement object & pre-fetch another;
2. Return next advertisement object-id only; and
3. Pre-fetch next advertisement object only.

By separating the function request (1) into its two components, (2) and (3), object interpreter 435 is now able to determine when to request advertisement object-id's and from its knowledge of the page build process, is able to best determine when another advertisement object can be pre-fetched, thus causing the least impact on the page response time. For example, by examining the PPT, object interpreter 435 may determine whether any object requests are outstanding. If there are out-

standing requests, advertisement request type would be used. When all requested objects are retrieved, object interpreter 435 then issues an advertisement request type 3. Alternatively, if there are no outstanding requests, object interpreter 435 issues an advertisement request type 1. This typically corresponds to the user's "think time" while examining the information presented and when RS 400 is in the Wait for Event state (D).

Data collection manager 441 is invoked by object interpreter 435 and keyboard manger 434 to keep records about what objects a user has obtained (and, if a presentation data segment 530 is present, seen) and what actions users have taken (e.g. "NEXT," "BACK," "LOOK," etc.).

The data collection events that are to be reported during the user's session are sensitized during the logon process. The logon response message carries a data collection indicator with bit flags set to "on" for the events to be reported. These bit flags are enabled (on) or disabled (off) for each user based on information contained in the user's profile stored and sent from high function host 110. A user's data collection indicator is valid for the duration of his session. The type of events to be reported can be changed at will in the host data collection application. However, such changes will affect only users who logon after the change.

Data collection manager 441 gathers information concerning a user's individual system usage characteristics. The types of informational services accessed, transactions processed, time information between various events, and the like are collected by data collection manager 441, which compiles the information into message packets (not shown). The message packets are sent to network 10 via object/communication manager interface 443 and link communications manager 444. Message packets are then stored by high function host 110 and sent to an offline processing facility for processing. The characteristics of users are ultimately used as a means to select or target various display objects, such as advertisement objects, to be sent to particular users based on consumer marketing strategies, or the like, and for system optimization.

Object/communications manager interface 443 is responsible for sending and receiving DIA (Data Interchange Architecture described above) formatted messages to or from interactive network 10. Object/communications manager 443 also handles the receipt of objects, builds a DIA header for messages being sent and removes the header from received DIA messages or objects, correlates requests and responses, and guarantees proper block sequencing. Object/communications manager interface 443 interacts with other native code modules as follows: object/communications manager 443 (1) receives all RS 400 object requests from object storage facility 439, and forwards objects received from network 10 via link communications manager 444 directly to the requesting modules; (2) receives ad list requests from ad manager 442, which thereafter periodically calls object/communications manager 443 to receive ad list responses; (3) receives data collection messages and send requests from data collection manager 441; (4) receives application-level requests from TBOL interpreter 438, which also periodically calls object/communications manager interface 443 to receive responses (if required); and (5) receives and sends DIA formatted objects and messages from and to link communications manager 444.

Object/communications manager interface 443 sends and receives DIA formatted messages on behalf of TBOL interpreter 438 and sends object requests and receives objects on behalf of object storage facility 439. Communication packets received containing parts of requested objects are passed to object storage facility 439 which assembles the packets into the object before storing it. If the object was requested by object interpreter 435, all packets received by object storage facility 439 are also passed to object interpreter 435 avoiding the delay required to receive an entire object before processing the object. Objects which are pre-fetched are stored by object storage facility 439.

Messages sent to interactive network 10 are directed via DIA to applications in network 10. Messages may include transaction requests for records or additional processing of records or may include records from a partitioned application program object or data collection manager 441. Messages to be received from network 10 usually comprise records requested in a previous message sent to network 10. Requests received from object storage facility 439 include requests for objects from storage in interactive system 10. Responses to object requests contain either the requested object or an error code indicating an error condition.

Object/communications manager 443 is normally the exclusive native code module to interface with link communications manager 444 (except in the rare instance of a fatal error). Link communications manager 444 controls the connecting and disconnecting of the telephone line, telephone dialing, and communications link data protocol. Link communications manager 444 accesses network 10 by means of a communications medium (not shown) link communications manager 444, which is responsible for a dial-up link on the public switched telephone network (PSTN). Alternatively, other communications means, such as cable television or broadcast media, may be used. Link communications manager 444 interfaces with TBOL interpreter for connect and disconnect, and with interactive network 10 for send and receive.

Link communications manager 444 is subdivided into modem control and protocol handler units. Modem control (a software function well known to the art) hands the modem specific handshaking that occurs during connect and disconnect. Protocol handler is responsible for transmission and receipt of data packets using the TCS (TRINTEX Communications Subsystem) protocol (which is a variety of OSI link level protocol, also well known to the art).

Fatal error manager 469 is invoked by all reception system components upon the occurrence of any condition which precludes recovery. Fatal error manager 469 displays a screen to the user with a textual message and an error code through display manager 461. Fatal error manager 469 sends an error report message through the link communications manager 444 to a subsystem of interactive network 10.

The source code for RS 400 is provided as part of this specification. This source code can be found in the application file and is incorporated herein by reference. Nomenclature for the various service software 430 modules may differ, but the functions described herein are implemented in the source code. Some functions described herein are implemented across modules in source code. The following is a concordance of the terms used in this section of the disclosure and the terms used in the source code:

Specification	Source Code
Keyboard Manager =	Input Manager/Event Processor
Object Interpreter =	Object Processor
TBOL Interpreter =	API or Logic Interpreter (the above 3 modules are referred to as the Service Manager)
Object Storage Facility =	Object Manager
Object/Communications Manager =	Message Manager and Communications Manager
Ad Manager =	Ad Manager
Display Manager =	Display Manager
Data Collection Manager =	Data Collection Manager
Link Communications Manager =	Communications Manager

The source code for the reception system 400 software is provided in the accompanying volumes 1 to 5, wherein the volume pages are consecutively numbered in accordance with the respective directories and sub-directories for source code files which are as follows:

Directory	Subdirectory	Subdirectory
rs	api	c inc
rs	applib	asm c inc
	cm	asm c inc
	esp	asm c inc
	los	asm c inc
	oversutl	c inc
	rsk	asm c inc
	sm	asm c inc
	storeutl	c inc
	tmk	asm c inc
	ver__esp	c inc
	ver__over	c inc
	ver__sm	c inc
	ver__stor	c inc

SAMPLE APPLICATION
The page illustrated in FIG. 3(b) corresponds to a partitioned application that permit's a personal com-

puter user to purchase apples. It shows how the monitor screen 414 of personal computer 405 might appear to the user. The displayed page includes a number of page partitions and corresponding page elements.

The PTO 500 representing this page 255 is illustrated in FIG. 9. PTO 500 defines the composition of the page, including header 250, body 260, display field 270, 271, 272, advertisement 280, and command bar 290. PEOs 504 shown in FIG. 9 are associated with page partitions numbered; e.g., 250, 260, 280. They respectively, present information in the header 250, identifying the page topic as ABC APPLES; in the body 260, identifying the cost of apples; and prompt the user to input into fields within body 260 the desired number of apples to be ordered. In advertisement 280, presentation data and a field representing a post-processor that will cause the user to navigate to a targetable advertisement, is presented.

In FIG. 9, the structure of the PTO 500 can be traced. PTO 500 contains a page format call segment 526, which calls Page Format Object (PFO) PFO 502 describes the location and size of partitions on the page and numbers assigned to each partition. The partition number is used in page element call segments 522 so that an association is established between a called PEO 504 and the page partition where it is to be displayed. Programs attached to this PEO can be executed only when the cursor is in the page partition designated within the PEO.

PTO 500 contains two page element call segments 522, which reference the PEOs 504 for partitions 250 and 260. Each PEO 504 defines the contents of the partition. The header in partition 250 has only a presentation data segment(s) 530 in its PEO 504. No input, action, or display fields are associated with that partition.

The PEO 504 for partition 260 contains a presentation data segment 530 and field definition segments 516 for the three fields that are defined in that partition. Two of the fields will be used for display only. One field will be used for input of user supplied data.

In the example application, the PEO 504 for body partition 260 specifies that two program objects 508 are part of the body partition. The first program, shown in Display field 270, 271, 272, is called an initializer and is invoked unconditionally by TBOL interpreter 438 concurrently with the display of presentation data for the partition. In this application, the function of the initializer is represented by the following pseudo-code:

1. Move default values to input and display fields;
2. "SEND" a transaction to the apple application that is resident on interactive system 10;
3. "RECEIVE" the result from interactive system 10; i.e. the current price of an apple;
4. Move the price of an apple to PEV 271 so that it will be displayed;
5. Position the cursor on the input field; and
6. Terminate execution of this logic.

The second program object 508 is a field Post-processor. It will be invoked conditionally, depending upon the user keystroke input. In this example, it will be invoked if the user changes the input field contents by entering a number. The pseudo code for this post-processor is as follows:

1. Use the value in PEV 270 (the value associated with the data entered by the user into the second input data field 270) to be the number of apples ordered.

5,347,632

97

2. Multiply the number of apples ordered times the cost per apple previously obtained by the initializer;

3. Construct a string that contains the message "THE COST OF THE APPLES YOU ORDERED IS \$45.34;";

4. Move the string into PEV 272 so that the result will be displayed for the user; and

5. Terminate execution of this logic.

The process by which the "APPLES" application is displayed, initialized, and run is as follows.

The "APPLES" application is initiated when the user navigates from the previous partitioned application, with the navigation target being the object id of the "APPLES" PTO 500 (that is, object id ABC1). This event causes keyboard manager 434 to pass the PTO object id, ABC1 (which may, for example, have been called by the keyword navigation segment 520 within a PEO 504 of the previous partitioned application), to object interpreter 435. With reference to the RS application protocol depicted in FIG. 6, when the partitioned application is initiated, RS 400 enters the Process Object state (B) using transition (1). Object interpreter 435 then sends a synchronous request for the PTO 500 specified in the navigation event to object storage facility 439. Object storage facility 439 attempts to acquire the requested object from local store 440 or from delivery system 20 by means of object/communication manager 443, and returns an error code if the object cannot be acquired.

Once the PTO 500 is acquired by object/communications manager 443, object interpreter 435 begins to build PPT by parsing PTO 500 into its constituent segment calls to pages and page elements, as shown in FIG. 4d and interpreting such segments. PFO and PEO call segments 526 and 522 require the acquisition of the corresponding objects with object id's <ABCF>, <ABCX> and <ABCY>. Parsing and interpretation of object ABCY requires the further acquisition of program objects <ABCI> and <ABCJ>.

During the interpretation of the PEOs 504 for partitions 250 and 260, other RS 400 events are triggered. This corresponds to transition (2) to interpret pre-processors state (C) in FIG. 6. Presentation data 530 is sent to display manager 461 for display using a NAPLPS decoder within display manager 461, and, as the PEO <ABCY> for partition 260 is parsed and interpreted by object interpreter 435, parameters in program call segment 532 identify the program object <ABCI> as an initializer. Object interpreter 435 obtains the program object from object storage facility 439, and makes a request to TBOL interpreter 438 to execute the initializer program object 508 <ABCI>. The initializer performs the operations specified above using facilities of the RS virtual machine. TBOL interpreter 438, using operating environment 450, executes initializer program object 506 <ABCI>, and may, if a further program object 508 is required in the execution of the initializer, make a synchronous application level object request to object storage facility 439. When the initializer terminates, control is returned to object interpreter 435, shown as the return path in transition (2) in FIG.

Having returned to the process object state (B), object processor 435 continues processing the objects associated with PTO <ABCI>. Object interpreter continues to construct the PPT, providing RS 400 with an environment for subsequent processing of the PTO <ABCI> by pre-processors and post-processors at the

98

page, partition, and field levels. When the PPT has been constructed and the initializer executed, control is returned to keyboard manager 434, and the RS enters the wait for event (E) State, via transition (4), as shown in FIG. 6.

In the wait for event state, the partitioned application waits for the user to create an event. In any partitioned application, the user has many options. For example, the user may move the cursor to the "JUMP" field 296 on the command bar 290, which is outside the current application, and thus cause subsequent navigation to another application. For purposes of this example, it is assumed that the user enters the number of apples he wishes to order by entering a digit in display field 271.

Keyboard manager 434 translates the input from the user's keyboard to a logical representation independent of any type of personal computer. Keyboard manager 434 saves the data entered by the user in a buffer associated with the current field defined by the location of the cursor. The buffer is indexed by its PEV number, which is the same as the field number assigned to it during the formation of the page element. Keyboard manager 434 determines for each keystroke whether the keystroke corresponds to an input event or to an action or completion event. Input events are logical keystrokes and are sent by keyboard manager to display manager 461, which displays the data at the input field location. Display manager 461 also has access to the field buffer as indexed by its PEV number.

The input data are available to TBOL interpreter 438 for subsequent processing. When the cursor is in a partition, only the PEVs for that partition are accessible to the RS virtual machine. After the input from the user is complete (as indicated by a user action such as pressing the RETURN key or entry of data into a field with an action attribute), RS 400 enters the Process Event state (E) via transition (4).

For purposes of this example, let us assume that the user enters the digit "5" in input field 270. A transition is made to the process event state (E). Keyboard manager 434 and display manager 437 perform a number of actions, such as the display of the keystroke on the screen, the collection of the keystroke for input, and optionally, the validation of the keystroke, i.e. numeric input only in numeric fields. When the keystroke is processed, a return is made to the wait for event state (D). Edit attributes are specified in the field definition segment.

Suppose the user inputs a "6" next. A transition occurs to the PE state and after the "6" is processed, the Wait for Event (D) state is reentered. If the user hits the "completion" key (e.g., ENTER) the Process Event (E) state will be entered. The action attributes associated with field 272 identify this as a system event to trigger post-processor program object <ABCJ>. When the interpretive execution of program object <ABCJ> is complete, the wait for event state (D) will again be entered. The user is then free to enter another value in the input field, or select a command bar function and exit the apples application.

While this invention has been described in its preferred form, it will be appreciated that changes may be made in the form, construction, procedure and arrangement of its various elements and steps without departing from its spirit or scope.

What is claimed is:

1. A reception system provided in an interactive computer network, the reception system for presenting par-

tioned applications that include informational and transactional services to a user, the reception system comprising:

input means for receiving user inputs, at least some of which include requests for partitioned applications; storage means for storing objects, the objects collectively including data and executable program instructions used in generating the partitioned applications, and the storage means further retaining objects between requests for partitioned applications;

object processing means responsive to the input means for selectively retrieving and interpreting objects to extract data and program instructions required for composing and generating the partitioned applications; and

communication means for sending object requests arising within the object processing means to and receiving objects from the interactive network when objects required for generating the partitioned applications are unavailable at the storage means.

2. The reception system according to claim 1, wherein the partitioned applications and user inputs are processed according to a protocol provided by the reception system.

3. The reception system according to claim 1, wherein the object processing means includes elements for interpreting objects having a prescribed structure that includes one or more embedded objects.

4. The reception system according to claim 1, wherein the object processing means includes elements for interpreting objects having a prescribed structure that includes one or more embedded calls to other objects.

5. The reception system according to claim 1, wherein the object processing means includes elements for interpreting objects having a prescribed structure that includes one or more embedded objects and one or more embedded calls to other objects.

6. The reception system according to claim 1, wherein the object processing means includes elements for interpreting objects having a prescribed structure that includes no embedded objects and no embedded calls to other objects.

7. The reception system according to claims 3, 4, 5 or 6, wherein the object processing means includes elements for interpreting an object structure including a header and one or more segments wherein each segment has a prescribed structure.

8. The reception system according to claim 7, wherein the header is extendable.

9. The reception system according to claim 7, wherein the object processing means includes elements for interpreting an object structure in which the segment structure identifies segment length and type.

10. The reception system according to claim 7, wherein the object processing means includes elements for interpreting an object structure in which the header includes an object identifier.

11. The reception system according to claim 10, wherein the object processing means includes elements for interpreting an object identifier that includes an object space identifier for designating an object address space.

12. The reception system according to claim 10, wherein the object processing means includes elements for interpreting an object identifier that includes a set

identifier for designating an object set within an object address space.

13. The reception system according to claim 10, wherein the object processing means includes elements for interpreting an object identifier that includes an occurrence field for designating an object within an object set.

14. The reception system according to claim 10, wherein the object processing means includes elements for interpreting an object identifier that includes a type field for designating the use of the object and the structure of the object identifier.

15. The reception system according to claim 7, wherein the object storage means includes elements for interpreting an object structure including a header having control attributes for indicating the permanency and currency of the object.

16. The reception system according to claim 15, wherein the storage means includes elements for selectively storing objects between user sessions according to the control attributes of the respective objects.

17. The reception system according to claim 15, wherein the storage means includes elements for selectively storing objects during user sessions according to the control attributes of the respective objects.

18. The reception system according to claim 7, wherein the object processing means includes elements for interpreting an object structure in which the header includes means for indicating the length of the object.

19. The reception system according to claim 1, wherein the communication means is adapted for sending messages to and receiving messages from the interactive network.

20. The reception system according to claim 19, wherein the data storage means communicates with the communication means, for requesting a desired object from the interactive network if the desired object is not present in the storage means.

21. The reception system according to claim 1, further including collection means for collecting and storing data concerning object usage at the reception system.

22. The reception system according to claim 21, wherein the reception system includes a display and wherein advertisements are selectively exhibited at the display in response to the object usage data assembled by the collection means.

23. The reception system according to claim 1, wherein the input means includes input management means for translating the user inputs into a personal computer independent format.

24. The reception system accordingly to claim 1, wherein the object processing means includes elements for identifying objects for interpretation that are obtained from the interactive network in response to predetermined initial parameters.

25. The reception system according to claim 1, wherein the storage means includes a random access memory.

26. The reception system according to claim 1, wherein the storage means includes a diskette or other magnetic media.

27. The reception system according to claim 1, wherein the storage means includes optical medium.

28. The reception system according to claim 1, wherein the storage means includes a broadcast medium.

101

29. A reception system provided in an interactive computer network, the reception system for presenting partitioned applications including informational and transactional services to a user, the reception system comprising:

input means for receiving user inputs, at least some of which may include requests for partitioned applications, and at least some of which may include messages;

storage means for storing objects, the objects collectively including data and executable program instructions used in generating the partitioned applications, and the storage means further retaining objects between requests for partitioned applications;

objects processing means responsive to the input means for selectively retrieving and interpreting objects to extract data and program instructions required for composing and generating the partitioned applications;

communication means for passing messages arising at the input means and object requests arising at the object processing means to and receiving objects and messages from the interactive network;

collection means in communication with the input means and the communication means for compiling object use data and passing the compiled object use data to the interactive network.

30. The reception system according to claim 29, further including advertisement management means for pre-fetching advertisement objects from the interactive network, each of which advertisement objects includes an object identifier, and controlling presentation of advertisements associated with the advertisement objects in response to the compiled object use data.

31. The reception system according to claim 30, wherein the advertisement management means includes an advertisement queue for storing the object identifiers of the advertisement objects for the purpose of pre-fetching the advertisement objects.

32. The reception system according to claim 31, wherein the advertisement queue can store a variable number of the object identifiers based on predetermined parameters.

33. The reception system according to claim 29, wherein the compiled object use data is processed by the interactive network.

34. Method for operating a compute as a reception system for presenting partitioned applications to a user, the partitioned application being made up of objects that collectively include data and program instructions, the method comprising the steps of:

a. receiving requests for partitioned applications at the reception system;

b. interpreting objects to extract data and program instructions required for composing and generating the requested partitioned applications;

c. executing program instructions that may be included within the objects for which the requested partitioned applications can be generated;

d. storing objects from which the requested partitioned applications can be generated and retaining

5,347,632

102

objects between requests for partitioned applications;

e. communicating with the network to obtain objects from which the requested partitioned applications can be generated that are not available at the reception system;

f. causing the interpreted object data to be presented at the reception system;

wherein, when a partitioned application is requested, the reception system determines the objects required to be executed for generating the partitioned application; determines whether the required objects are available at the reception system; secures required objects not available at the reception system from the network; and interprets the required objects to obtain the data and program instructions required for composing and presenting the partitioned application, and presents the application by supplying the necessary data for presentation.

35. The method of claim 34 wherein the receiving of requests for partitioned applications includes transforming requests for partitioned applications entered at the reception system computer into logical events which are interpreted so that required objects for the application can be identified and organized.

36. The method of claim 35 wherein interpreting the objects includes creating a page processing table to control collection of objects required to be executed for presenting the requested application.

37. The method of claim 36 wherein storing and retaining the objects includes monitoring objects required for a partitioned application to assure that the most current version for each of the objects is provided for application presentation.

38. The method of claim 37 further including collecting data regarding the frequency of use of various objects required for the partitioned applications requested.

39. The method of claim 37 further including providing advertisement objects to the reception system for presenting advertisements as part of a partitioned application.

40. The method of claim 34 wherein communication with the network includes communicating objects and messages.

41. The method of claim 34 wherein executing program instructions includes executing object program instructions prior to execution of the objects containing data in order to effect the collection of objects for presentation, and further includes executing object program instructions following presentation of data to undertake action in response to the presented data.

42. The method of claim 34 wherein the reception system undertake multitasking of events relating to presentation of partitioned applications in cooperation with the operating system running at the computer.

43. The method of claim 34 wherein interpreting objects includes interpreting the objects by parsing the objects into segments according to a prescribed structure for the objects.

* * * * *

United States Patent [19]

Gorog

[11] Patent Number: **4,947,028**
 [45] Date of Patent: **Aug. 7, 1990**

- [54] **AUTOMATED ORDER AND PAYMENT SYSTEM**
 [75] Inventor: **Jonathan M. Gorog, Falls Church, Va.**
 [73] Assignee: **Arbor International, Inc., Vienna, Va.**
 [21] Appl. No.: **221,536**
 [22] Filed: **Jul. 19, 1988**
 [51] Int. Cl.⁵ **G06F 1/08**
 [52] U.S. Cl. **235/381; 235/380; 235/383; 235/472**
 [58] Field of Search **235/380, 383, 381, 472; 902/22**

[56] References Cited

U.S. PATENT DOCUMENTS

3,292,489	8/1964	Johnson et al.	88/24
3,668,312	6/1972	Yamamoto et al.	178/6.8
4,115,870	9/1978	Lowell	364/900
4,329,684	5/1982	Monteath et al.	340/707
4,415,065	11/1983	Sanostedt	186/39
4,471,218	9/1984	Culp	235/472
4,516,016	5/1985	Kodron	235/472
4,525,624	6/1985	Pontefract	235/383
4,578,572	3/1986	Hice	235/472
4,608,487	8/1986	Awane et al.	235/383
4,621,189	9/1986	Kumar et al.	235/472
4,621,259	9/1986	Schepers et al.	340/707
4,707,592	11/1987	Warr	902/22
4,752,676	6/1988	Leonard	235/493
4,812,628	3/1989	Boston	235/280
4,812,629	3/1989	O'Neil	235/383

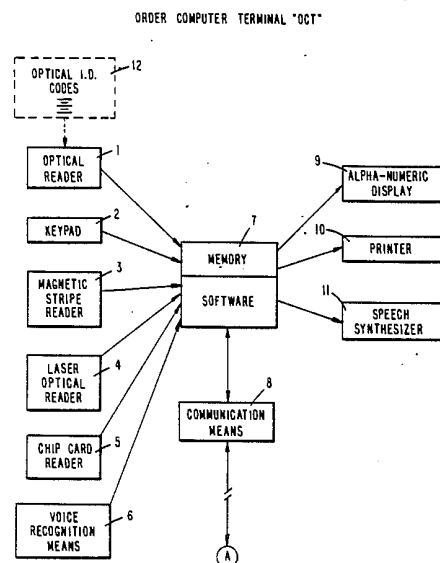
Primary Examiner—Harold I. Pitts
 Attorney, Agent, or Firm—Jon L. Roberts

[57] ABSTRACT

An automated order and payment system for use by consumers to rapidly order products and services from

any location at which the consumer is present at the time of ordering. The system receives information about the products/services to be ordered by means of signals generated by scanning identification codes imprinted in advertising media or displayed to a consumer on a television screen. A special version of the invention is modified to accept voice command via a voice recognition means for those physically handicapped persons unable to perform manual data entry tasks. The consumer uses an optical scanning means embodied in the Order Computer Terminal to scan identification code associated with a company and identification codes associated with the products/services desired. This product and company information is stored in the Order Computer Terminal along with credit information retrieved from a plurality of storage means used on credit cards and subsequently transmitted when desired by the consumer to a Central Computer System. The Central Computer System simultaneously receives information from multiple order computer terminals and verifies that the products or services from the desired company are in fact available. The Central Computer System also verifies the credit worthiness of the consumer by searching for such information from credit data bases. When the Central Computer System determines that the desired products/services are available and that the consumer is credit worthy, an order verification signal is sent to the individual consumer's order computer terminal whereupon the consumer verifies that he/she wishes to order the products/services communicated to the central computer system. Once the consumer verifies the order, the automated order and payment system places the order for the products/services desired and provides the appropriate credit reference to the supplier of the product/service. The automated order and payment system capabilities are more fully set forth herein.

14 Claims, 5 Drawing Sheets



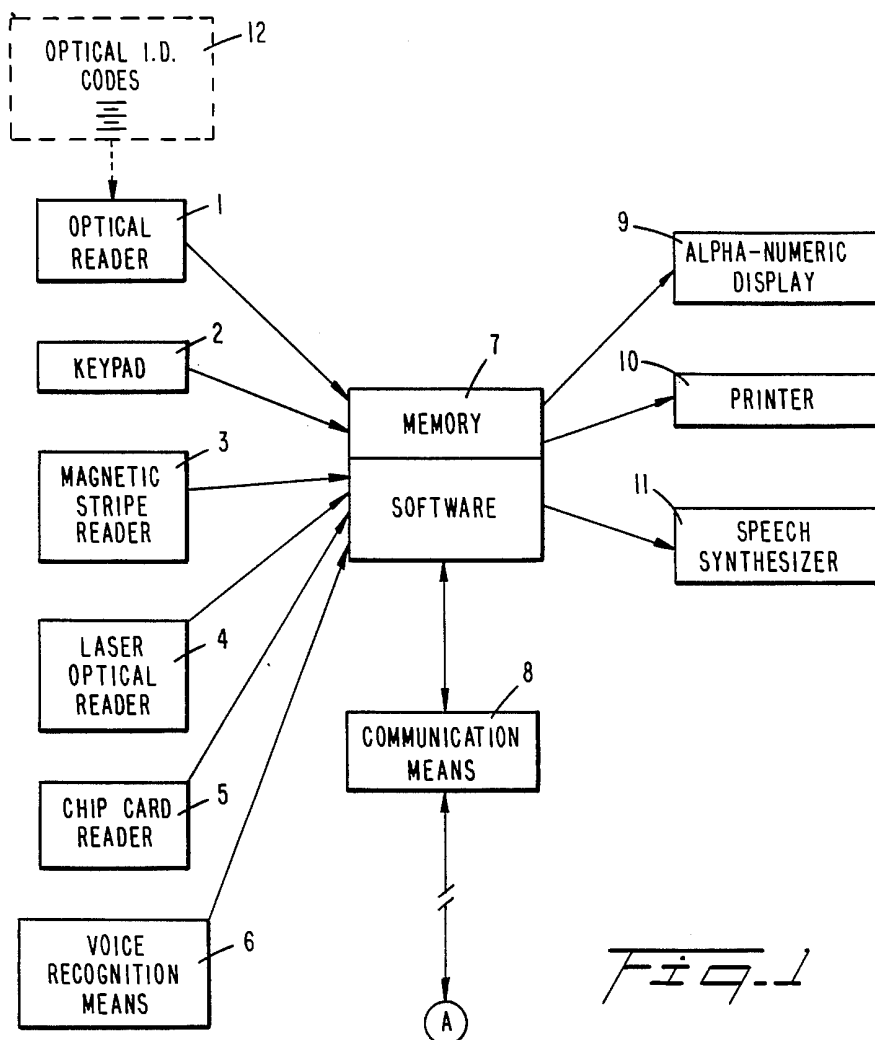
U.S. Patent

Aug. 7, 1990

Sheet 1 of 5

4,947,028

ORDER COMPUTER TERMINAL "OCT"



U.S. Patent

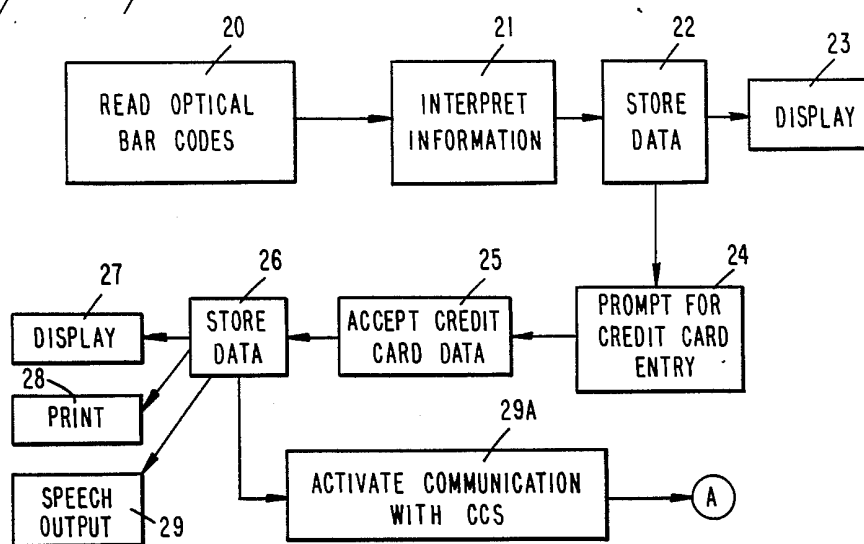
Aug. 7, 1990

Sheet 2 of 5

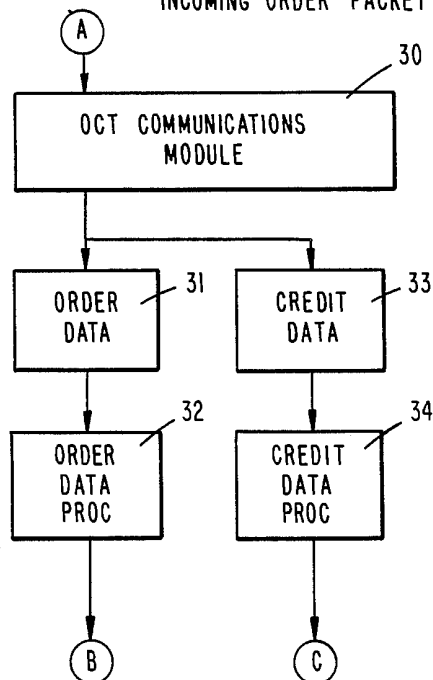
4,947,028

Fig. 2

OCT PROCESS



INCOMING ORDER PACKET PROCESSING

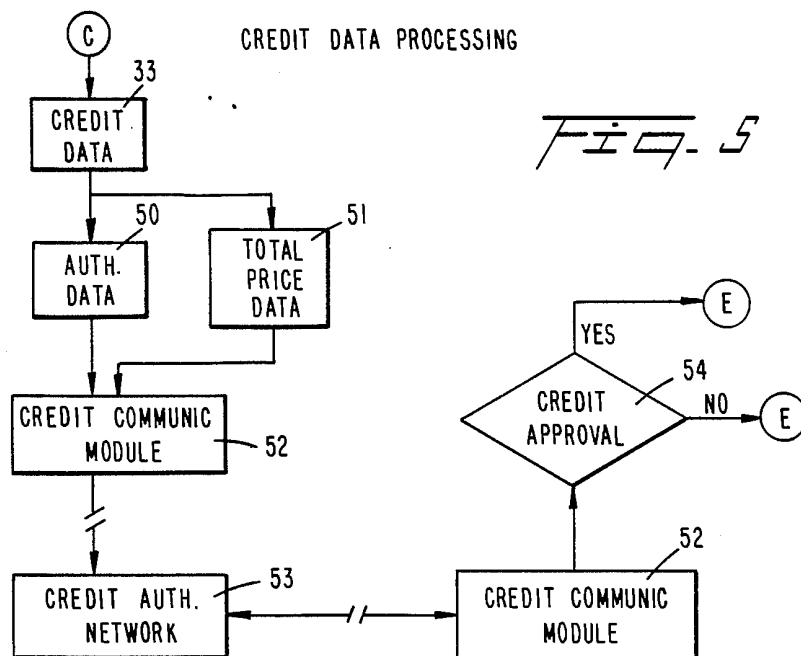
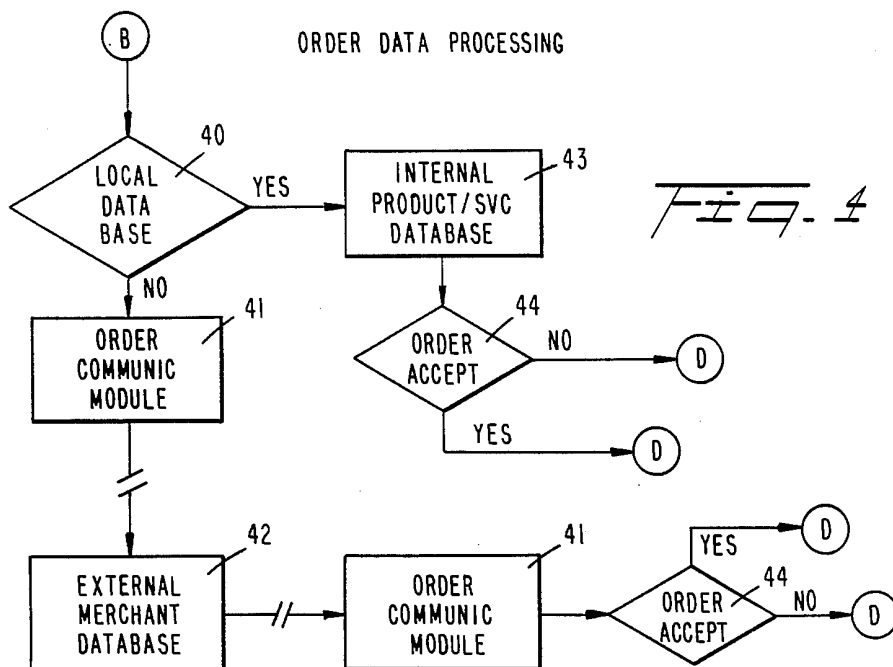
*Fig. 3*

U.S. Patent

Aug. 7, 1990

Sheet 3 of 5

4,947,028



U.S. Patent

Aug. 7, 1990

Sheet 4 of 5

4,947,028

CSS ORDER ACCEPTANCE PROCESS

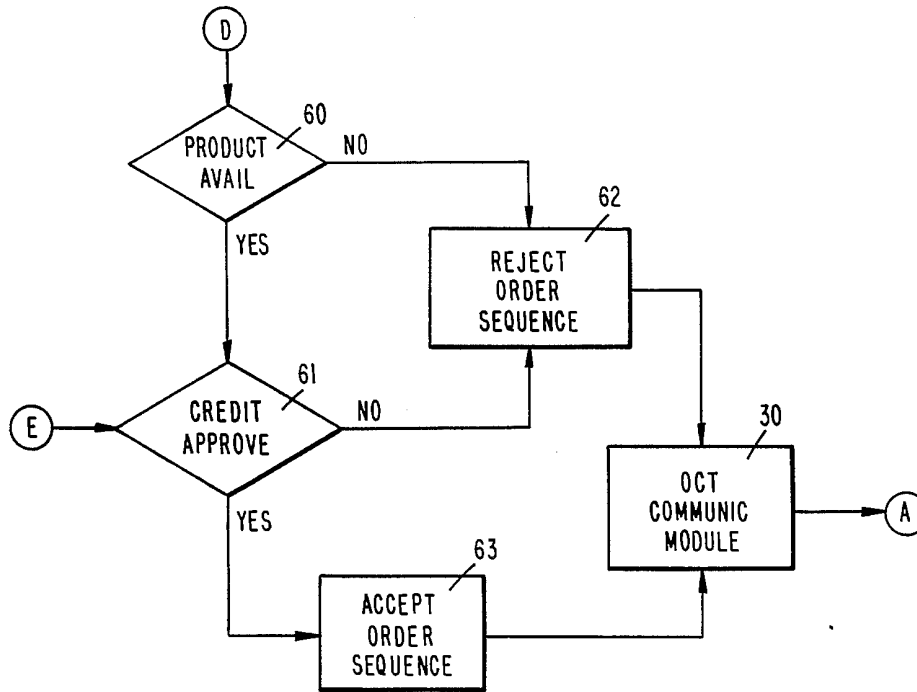


FIG. 6

U.S. Patent

Aug. 7, 1990

Sheet 5 of 5

4,947,028

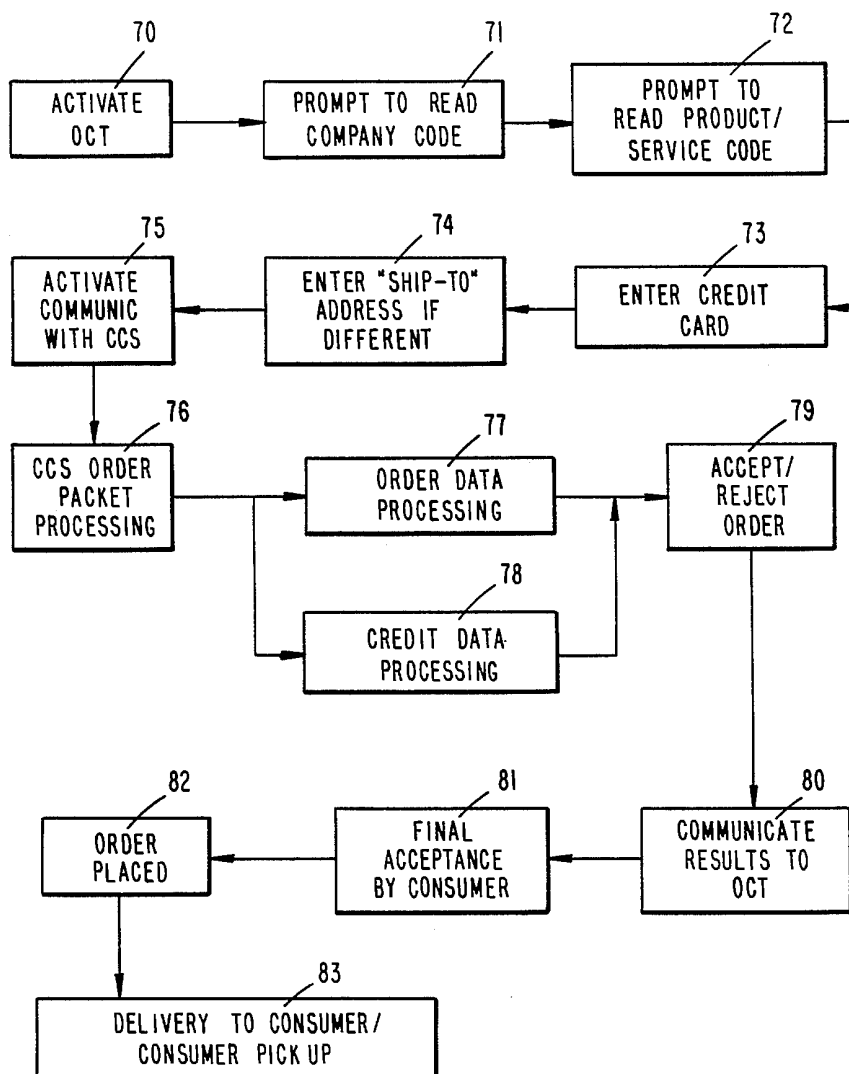
AUTOMATED ORDER AND
PAYMENT SYSTEM PROCESS
FLOW

Fig. 7

4,947,028

1

AUTOMATED ORDER AND PAYMENT SYSTEM**BACKGROUND OF THE INVENTION****1. Field of the Invention**

This invention relates generally to automated order entry systems and more specifically to electronic communication of orders from a buyer to a seller of products/services.

The originality of the invention lies in the integration of existing devices, products and networks to accomplish a unique service which will make the process of buying and selling significantly more efficient. By using electronic communication technologies, this invention will allow the general public, including the visually and mobility handicapped, to place orders, and to pay for and receive merchandise and services, directly from their domiciles. Additionally, the same process will be used for merchant to merchant ordering and sales transactions.

2. Background

Several types of electronic data entry systems are described in issued patents. These fall into several classes. Certain data entry systems are caused to record data by an operator pressing a series of keys to allow the operator to enter alphanumeric data. Other such systems cause data to be entered through use of a manually scanned optical sensor. One such combination unit having both keyboard input and optical scanner input is described in Kumar et al. U.S. Pat. No. 4,621,189. Similarly, Hice U.S. Pat. No. 4,578,572 discloses an apparatus for personal identification which comprises a bar code printer and separate bar code reading apparatus connected to a portable keyboard data input device.

Other bar code reading devices have also been disclosed in issued U.S. Patents. In Schepers et al. U.S. Pat. No. 4,621,259 an optical bar code reader connected to a station selecting apparatus in a television set allows a user to select one particular station for viewing. In Sandstedt U.S. Pat. No. 4,415,065, an apparatus is disclosed containing an optical bar code reader attached to a portable micro-processor for rapid entry of orders in a restaurant or a retail vending facility. In Kodron U.S. Pat. No. 4,516,016, an apparatus is disclosed that is very similar to the Sandstedt invention for use in recording orders in restaurants that contains an optical bar code reader that is connected to a central processing unit for entry of restaurant orders. In Awane et al. U.S. Pat. No. 4,608,487, a bar code reader is used to input information to automated vending machines. In Culp U.S. Pat. No. 4,471,218, a portable data entry terminal is disclosed containing an optical bar code reader and memory that subsequently transfers the data from the portable unit to a central computer.

Optical readers known as "light pens" have also been used to interact with cathode ray tubes ("CRT") to communicate with a central data base to indicate the selection of an option by a user. Yamamoto et al. U.S. Pat. No. 3,668,312, discloses a system whereby a party receiving a television image can use a light pen to indicate a selection of an option. This invention is used in the context of a telephone system. In Monteath et al. U.S. Pat. No. 4,329,684, a light sensing apparatus capable of sensing either a bar code or the light output in a particular area of a television screen to allow input of information to a central computer is disclosed. Johnson et al. U.S. Pat. No. 3,292,489, discloses a system for retrieving information from a database where data is

2

displayed on a CRT with an associated optical code which is scanned by a hand-held optical sensor, which in turn provides information to a database for retrieval of the data required.

Portable data storage devices are also disclosed in issued U.S. patents. In Pontefract U.S. Pat. No. 4,525,624, a data storage device is disclosed that stores information for a salesman, while information is input via a key pad. At the close of each day, the stored data is transmitted via telephone to a central computer. Similarly, in Lowell U.S. Pat. No. 4,115,870, a hand-held data processing terminal is disclosed that stores data input via a key pad, which device also contains a data transmission circuit to allow information stored to be transmitted over telephone lines to a central computer. In summary, these patents all describe systems, each of which serves only a part of the entire retail cycle of customer demand, supplier filling that demand, payment for goods or services desired and delivery of those goods and services.

Many of the above patents describe systems which collect data and transfer data to a central data base. None of the patents describe data communication back from the central database to the data collection device in the same process. Additionally, none of the patents address the needs of the visually and mobility handicapped segment of the general public who cannot operate the devices described. Finally, none of the above patents describe a means to account for the credit needs of the public which are an integral part of the vast majority of the purchases made by consumers today.

SUMMARY OF THE INVENTION

The Automated Order and Payment System of this invention allows consumer transactions for goods and services to take place in a faster and more efficient manner than currently available thereby reducing the cost of selling such goods to consumers. The invention provides a simplified ordering system that eliminates the need for any order form to be filled out by the consumer. It further does not require the consumer to go to the place of sale in order to obtain the merchandise desired.

The system comprises three major components:

- (a) A central computer system ("CCS") with a variety of programs, processing and storage capability and communications capabilities to allow input and output communications with order computer terminals.
- (b) A product/service identification system consisting of symbols that are presented to the public via print and/or television media.
- (c) A order computer terminal ("OCT") with means to input data orally, optically, magnetically, electronically, and manually having associated order processing software and communications capabilities allowing receipt of communications from the CCS and further providing output communications to the CCS.

The CCS can send data to or receive data from the OCT's or from other computer systems, for the purpose of accepting data transmitted from such terminals or other computers over normal telephone lines, radio, television, satellite, or any other signals from remote locations to the CCS. The CCS can also communicate with other computers using accepted industry protocols.

3

4,947,028

4

The CCS has various computer software programs that allow product/service order information to be accepted and transmitted from the central computer. Such software will also confirm or deny orders for products based upon records of inventories that have been provided by participating businesses or by sending a query to other computers holding the necessary data records for participating businesses. The CCS also confirms/denies the payment medium chosen by the consumer by communicating with third party systems such as credit card authorization systems or individual businesses.

Upon completion of order acceptance and confirmation of payment media, the confirmed order with associated information is sent to the consumer via the communications output means of the CCS. The confirmation information is received by the OCT via its internal communication means and displayed or printed for the consumer. If the order or payment is denied, the CCS sends a denial message to the OCT from the CCS communication means, which message is received by the OCT communication means. Thus the OCT and the CCS communicate with each other via their two-way communications capability.

The invention operates through a series of identification codes that uniquely identify the company offering the product/service for sale and the individual product/service desired. Such identification codes may consist of, but are not limited to, bar codes which represent businesses and products/services being offered. These codes are printed in catalogs, magazines, newspapers and television advertisements, direct mail circulars, and any other medium that might communicate product information to consumers.

The OCT comprises data entry and storage and communication capabilities for use by the consumer. The consumer enters identifications data about the company from which he wishes to purchase products/services and data concerning the products/services to be purchased. This information is entered via an optical identification code reader which is manually scanned over an identification code. Information can also be entered manually via a key pad or orally via a voice recognition capability embodied in the OCT, which capability converts the human voice into digital data. The associated key pad or voice recognition capability also is used to input other information relating to the sale. The OCT also has the capability to read magnetic stripes, microchips, and optical storage media that are incorporated in credit cards using a plurality of means embodied in the OCT.

The OCT has a capacity to store orders from multiple companies and multiple orders from any given company.

A consumer uses the communications capacity of the OCT to transmit the order data via telephone lines, radio signal, or other communications links to the CCS. The OCT also has variable programming means such that it will prompt the user to enter data as required.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 illustrates the overall system structural arrangement.

FIGS. 2-7 illustrate various processing concepts within the system of FIG. 1.

DETAILED DESCRIPTION OF THE INVENTION

The automated order and payment system allows a consumer to place orders for goods and services through use of a order computer terminal ("OCT") that interacts with identification codes and voice commands that specifically identify companies and products/services, which codes appear in advertising media. The system allows rapid entry of orders for products and services without the consumer having to go to the location where such products and services are offered and with a minimum of consumer data entry. The invention consists of widely distributed OCTs and a central computer system ("CCS") that processes the orders and associated credit information that is sent to the CCS from the OCTs.

Through use of the OCTs, consumers rapidly enter information concerning the products and services desired with such information subsequently transmitted to the CCS for rapid processing and verification.

Referring first to FIG. 1, the OCT consists of an optical reader [1] that a consumer scans across identification codes [12] that may appear in magazine, newspaper, catalog, and televisions advertisements. These codes identify the company providing the product/service, the product/service desired and the price code for the product/service. Information sensed by the optical reader is placed in the OCT memory [7].

The OCT contains an alpha numerical key pad [2] by which a consumer can enter data when necessary and which contains a key for initiating the order process and a key for verifying an order once all data processing concerning that order has been completed.

For special applications where a visually or mobility handicapped person used the system, the OCT contains a human voice recognition means and may also contain a speech synthesizing means. The voice recognition means translates voice commands into digital data that is processed and stored in memory. The voice synthesizing means is used in lieu of the visual display thereby prompting the user through the ordering and payment sequence.

The OCT also contains a magnetic reading capability [3] that allows a consumer to use the magnetic stripe on a credit card to directly input the information contained therein to the order entry system. In this manner, data can be rapidly entered concerning the credit information of a consumer.

The OCT also contains a laser optical reader [4] that optically reads stored data that may be present on a credit card. Such optical data may also contain consumer, and other identifying data.

The OCT also contains means to read the data stored in microchips embedded in credit cards [5]. Data stored in microchips will also contain certain consumer identification and credit card information necessary to complete purchase transactions.

The OCT also contains means to recognize the speech of an individual in order to enter data orally [6] for those handicapped persons unable to manually use a credit card.

The OCT also contains a communications means [8]. This interface allows the OCT to communicate with the CCS via telephone or other communications media (i.e. optical fiber networks, radio frequency transmissions, and satellite communications means). In this way, the OCT can rapidly send information without error to the

4,947,028

5

CCS concerning the order information and information concerning the consumer's credit.

The OCT also has an alphanumeric display [9] that informs the consumer that the OCT is operating correctly, that the order has been accepted or rejected, and other information related to communication between the OCT and the CCS. Such information may also be printed via an integral printer means [10] or communicated to a handicapped individual via an integral speech synthesizing means [11].

Finally, the OCT contains both a memory capability and a capability to execute preprogrammed software [7]. This capability allows the OCT to store the software necessary to allow the OCT to interact with the consumer and the CCS. Further, the memory capability allows the OCT to store information presented to it by the optical reader, the key pad, the voice recognition system, the microchip reader, or the magnetic stripe reader which information relates to consumer identification data and to the goods or services desired for subsequent transmission via the communications interface to the CCS. Throughout this application, the credit and order data is referred to as the "order packet."

Referring to FIG. 2, The consumer uses the optical reading means of the OCT to read the optical identification codes [20] that are present in media advertising. The consumer first scans the identification code relating to the merchant or service provider. The consumer next scans the identification codes relating to the goods or services that the consumer desires to purchase.

In the case of a handicapped individual who cannot use the optical scanning means, the advertising data is input using the voice recognition means [6]. (Note: Throughout this specification, optical identification codes are exemplified as the input data. Data input capabilities of this inventions are in no sense so limited but should be viewed broadly. Indeed, it is a key aspect of this invention to provide alternative means of advertising and credit data input for those handicapped persons unable to handle a credit card or an optical identification code reader.)

In the situation where a handicapped consumer is ordering, that person will activate the OCT by a voice command and enter company data and product/service data orally. Company and product/service data will be presented to the handicapped public via media that has been customized to meet the specialized needs involved.

Software in the OCT interprets the optical identification code or orally input information [21] thereby obtaining company identification data, product or service identification data, and prices for the goods or services desired. After interpretation of this data, the information is stored [22] in the OCT memory and is either displayed [23] or confirmed by the voice synthesizer for the consumer.

After company, product and price data is entered, the OCT prompts the consumer to enter a credit card [24] to be used for the transaction. If handling a credit card is not possible for the handicapped person, credit data can be pre-stored in the OCT memory at the time the unit is provided to the handicapped person, thereby allowing such data to be transmitted without physically handling a credit card. The credit card data is accepted [25] and information is thereby acquired concerning the consumer's name, address, credit card number, type of credit card, and preferred delivery day, time and address for delivery. This information may be extracted from any of the various types of credit cards having

6

integral data storage capability (i.e. chip card, laser optical card, magnetic card).

Data extracted and interpreted from the consumer's credit card is stored in the OCT memory [26] and displayed for the consumer [27] on the OCT alphanumeric display or printed as desired [28] or verified orally via the speech synthesizing means [29].

After all data is stored, the consumer activates the OCT communications means [8, 29A] that allows the OCT to communicate with the CCS concerning the proposed sale of goods or services. The OCT memory now contains credit authorization information, total purchase price, company identification, a list of the products/services desired, the consumer's name and address, the delivery address, and the date and time for delivery.

Referring to FIG. 3, the CCS receives the order packets over a variety of transmission media (e.g., telephone line, optical fiber transmission lines, satellite data link) from OCTs via the OCT communications module [30]. This module contains the hardware and software necessary to receive order and credit information from OCTs when a consumer sends such information. The incoming order packet process causes the order packet data to be divided into order data [31] that is, the information relating to the merchant, identification of the goods or services, and the amount of items desired. This information then is subjected to the order data processing software [32] of the CCS.

In a similar fashion, the credit data [33] is separated from the incoming order packet information. This credit data contains the credit authorization data and other information relevant to the credit card in use, and the total price of the goods or services desired. This information is then subjected to the credit data process [34] of the CCS.

Referring next to FIG. 4, the CCS takes the order data information and determines if the order data may be further processed by using a local database internal to the CCS, or if the CCS must send such data to external merchant/supplier databases [40]. If the order information is not kept locally in the CCS, the CCS will activate the order communications module [41] which will in turn communicate with the external merchant-/supplier database [42]. Once the external database has determined that the goods or services are available, it communicates with the CCS order communications module notifying the CCS that the order has been accepted or not [44]. Such information is subsequently provided to the order acceptance process of the CCS (FIG. 6).

If the order data processing of the CCS determines that the product or service is represented in the CCS internal database of products and services [43], the CCS determines if the product of service is available in the desired quantity and/or at the desired time. The CCS verifies that inventory is present at the merchant/supplier, verifies the price of the goods or services desired, calculates the applicable tax, and confirms the delivery date and time desired. If the order information can be so fulfilled according to the database, the order accepted. If the database indicates the order cannot be satisfied it is rejected [44] and such information (acceptance or rejection) is communicated to the order acceptance process of the CCS (FIG. 6).

Referring to FIG. 5, credit data that is separated during the order packet process is sent to the credit processing software of the CCS. The credit data [33]

4,947,028

7

containing the information on the consumer, the type of credit card in use, the credit limits of the card (collectively the authorization data [50]), and the total purchase price [51] are divided into the data format necessary to communicate with the external credit database in question. The CCS credit communication module [52] allows this interaction with the external credit authorization network [53] to take place.

Once the external credit authorization network has reviewed the authorization and total price data, it communicates its determination back to the CCS via the credit communications module [52]. The CCS credit data processing software takes the incoming credit data and determines if the credit for the individual in question has been approved [54]. The results of this credit approval process are communicated to the CCS order acceptance process (FIG. 6).

Referring next to FIG. 6, the CCS order acceptance process makes two determinations. First, the CCS determines if products/services are available [60]. If not, the order is rejected [62] and the CCS so communicates to the OCT via the OCT communications module [30].

If the product or service is available, the CCS determines if credit for the purchase has been approved. If credit has not been approved, the order is rejected and the rejection is communicated to the OCT via the OCT communications module [30]. If credit has been approved, the CCS proceeds to the order acceptance sequence [63] and communicates the order acceptance and confirming information to the OCT via the OCT communications module [30].

Because the CCS contains internal databases of products and services, it also performs an inventory management service to those merchants who subscribe to the automated order and payment system. Thus an added capability of the invention is to provide merchants with inventory reports concerning their goods shipped and on hand as well as other inventory control features.

In summary, this process selects the merchant/supplier, confirms the availability of inventory to fulfill the sale, confirms the price, method of payment, and credit status of the consumer as well as the delivery date and method of delivery.

How To Use

When a consumer wants to place an order for products/services the consumer activates the OCT by pressing a function key on the OCT key pad. The consumer is then prompted through a series of steps that are displayed on the OCT display screen. These steps lead the consumer through the process required in order to complete an order. Referring to FIG. 7, these steps include but are not limited to the following:

1. Press the function key to activate the OCT [70].
2. The consumer is prompted to scan the optical reader over the company identification code printed or transmitted in the advertisement [71].
3. The consumer is next prompted to scan the optical reader over the product(s) identification code(s) listed in the advertisement and which the consumer desires to order [72].
4. The consumer is prompted to enter the form of payment using the payment input means of the terminal. For example the user is directed to pass a credit card through the magnetic stripe reader of the terminal [73].

It is important to note that step 4 is accomplished by the OCT reading such information from the credit card

8

in use. Such information will be present in the magnetic stripe, optical storage media or microchip on the credit card in use. It is an objective of this invention to provide a means of reading all such credit cards.

5. The consumer is next prompted to enter data concerning the address to which the consumer wishes the goods to be shipped if different from that stored on the credit card in use. This information is entered via the alphanumeric key pad of the OCT [74].
6. When this order entry process is completed, the consumer activates the communication capability of the OCT by causing the OCT to be connected to a telephone system or other transmission medium and depressing a "send" key [75]. At this point the OCT will dial a preprogrammed telephone number and transmit the order to the CCS. The information containing the order or orders, payment method, shipping addresses, and preferred method of delivery for each order (the order packet) is then processed by the CCS.
7. The CCS, via its internal software, accepts the order packet data [76], divides the data into order data [77] and credit data [78] performs the necessary analysis and communications to accept or reject the order [79], communicates the results to the OCT in question [80].
8. Once the order acceptance is communicated to the OCT, the consumer has a final opportunity to place the order or reject it [81] by so notifying the CCS via the OCT keypad input.
9. If the order is accepted by the consumer, it is placed by the CCS [82].
10. Once the order is placed, it is subsequently delivered to the consumer, or is filled for consumer pick-up on the date and at the time desired.

How to Use when Visually and/or Mobility-Handicapped

When a handicapped consumer wants to place an order for products/services, the consumer performs the following actions:

1. Activate the OCT by speaking an appropriate command into the OCT microphone [70].
2. The consumer is prompted by the OCT voice synthesizer to orally enter the company name that has been obtained from the advertising source that has been modified for use by the handicapped [71]. (e.g., Braille imprinting).
3. The consumer is prompted by the OCT voice synthesizer to orally enter the products/services and the price of those products and services desired [72].
4. The consumer is prompted by the OCT voice synthesizer to orally enter the form of payment using the payment input means of the OCT. For example, the user is prompted to pass a credit card through the magnetic stripe reader of the OCT [73]. If the consumer is unable to perform this task because of a particular handicap, the consumer will use the voice input means of the OCT to describe the payment method. Any special payments arrangements can be programmed into the OCT memory during the installation process.

The remaining operations of the product ordering and approval process proceed in the same fashion as previously described. Input and output during the ordering process can be achieved with any of the above

9

4,947,028

described input or output sources (i.e. printer, voice synthesizer, alphanumeric display).

While a specific embodiment of the invention has been discussed in the preceding sections, the invention should be viewed broadly and limited only by the scope of the appended claims.

What is claimed is:

1. An automated order and payment system which comprises:

A remote programmable data input/output means adapted to optically scan identification code information, and further adapted to accept credit card information obtained from the stored information on credit cards;

A communication means integral to said remote programmable data input/output means;

A memory means integral to such remote programmable data input/output and communication means that allows the storage of computer programs and information derived from printed or transmitted identification code information that has been optically scanned;

A central data processing means with communications capability adapted to receive information from a plurality of remote programmable data input/output means; and

Additional communication means allowing the remote data processing means to communicate with external data bases for credit authorization and product/service ordering purposes.

2. An automated order and payment system according to claim 1 wherein said remote data input/output means contains an optical reader for optically sensing identification codes, an alphanumeric key pad for data input, a communications interface to a central computer system which is an integral part of the automated order and payment system.

3. An automated order and payment system according to claim 2 wherein said remote programmable data input output device contains a magnetic reader to allow input of credit information from the magnetic stripe on credit cards.

4. An automated order and payment system according to claim 2 wherein said remote programmable data input output device contains a microchip reader to allow input of credit information from the microchips embedded in credit cards.

5. An automated order and payment system according to claim 2 wherein said remote programmable data input output device contains a laser optical data reader to allow input of credit information that is optically stored on credit cards.

6. An automated order and payment system according to claim 2 wherein said remote programmable input output device allows data entry via a speech processing means that allows oral instructions to be stored and used as digital data input.

7. An automated order and payment system according to claim 2 which comprises a central computer system which communicates with multiple remote programmable data input/output means over a plurality of transmission media.

8. An automated order and payment system according to claim 2 which receives information from the remote programmable data input/output means which data relates to company identification and product/ser-

10

vice identification for subsequent processing of orders for such products or services.

9. An automated order and payment system according to claim 2 which comprises a remote programmable data input/output means and central computer system which receives information concerning orders for goods and services and which also receives information concerning payment for such goods and services by means of credit and which central computer system obtains information from credit data bases concerning the credit worthiness of consumers ordering the products and services in question.

10. An automated order and payment system according to claim 9 which contains a data base of merchant/suppliers of products and services from which information is extracted concerning the availability of products and services that a consumer desires to order.

11. An automated order and payment system according to claim 10 which interacts with a consumer to notify the consumer of the acceptance or rejection of an order, or part of an order, based upon the approval/disapproval of credit and/or the availability/non-availability of the products or services desired.

12. A programmable remote order computer terminal ("OCT") comprising;

A remote programmable data input/output means adapted to optically scan identification code information, and further adapted to accept credit card information obtained from the stored information on credit cards;

A communication means integral to said remote programmable data input/output means;

A memory means integral to such remote programmable data input/output and communication means that allows the storage of computer programs and information derived printed or transmitted identification code information that has been optically scanned.

13. A process for rapidly ordering and paying for goods and services comprising;

The optical input by a consumer of a printed or transmitted identification codes used to identify merchants and goods/services desired by the consumer,

The entry of credit and consumer data by reading such information from a plurality of storage means resident on credit cards,

The storage and subsequent transmission of such data to a central computer systems,

The separate processing of such data by the central computer systems to determine (1) the availability of the product or service desired, and (2) the credit worthiness of the consumer ordering the products or services

The notification to the consumer of the approval or rejection of the sale desired

The delivery of the goods or products ordered by the consumer by prearranged parameters or selectively designated by the consumer at the time of sale.

14. A process for automated ordering of products and/or services according to claim 13 wherein the consumer and credit information is derived from a microchip credit card, magnetic storage credit card and/or optical data storage credit card.

* * * * *

REEXAMINATION CERTIFICATE (2037th)

[11] **B1 4,947,028**

Jun. 8, 1993

- | | | |
|-----------|--------|------------------|
| 4,752,676 | 6/1988 | Leonard et al. . |
| 4,812,628 | 3/1989 | Boston et al. . |
| 4,812,629 | 3/1989 | O'Neil et al. . |

Primary Examiner—Harold Pitts

[57] **ABSTRACT**

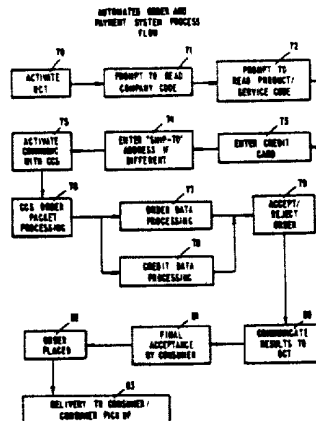
No. 90/002,536, Dec. 6, 1991

Patent No.: 4,947,028
 Issued: Aug. 7, 1990
 Appl. No.: 221,536
 Filed: Jul. 19, 1988

- ## [56] References Cited

3,292,489	7/1964	Johnson et al. .	
3,668,312	6/1972	Yamamoto et al. .	
3,719,927	3/1973	Michels et al. .	
4,115,870	9/1978	Lowell .	
4,266,271	5/1981	Chamoff et al. .	
4,277,837	7/1981	Stuckert .	
4,329,684	5/1982	Monteath et al. .	
4,341,951	7/1982	Benton .	
4,361,851	11/1982	Asip et al. .	
4,415,065	11/1983	Sandstedt .	
4,460,965	7/1984	Trehn et al. .	
4,471,218	9/1984	Culp .	
4,511,970	4/1985	Okano et al. .	
4,516,016	5/1985	Kodron .	
4,525,624	7/1985	Pontefract .	
4,578,572	3/1986	Hice .	
4,608,487	8/1986	Awane et al. .	
4,621,189	11/1986	Kumar et al. .	
4,621,259	11/1986	Schepers et al. .	
4,707,592	11/1987	Ware .	
4,734,858	3/1988	Schlaflly	364/408

An automated order and payment system for use by consumers to rapidly order products and services from any location at which the consumer is present at the time of ordering. The system receives information about the products/services to be ordered by means of signals generated by scanning identification codes imprinted in advertising media or displayed to a consumer on a television screen. A special version of the invention is modified to accept voice command via a voice recognition means for those physically handicapped persons unable to perform manual data entry tasks. The consumer uses an optical scanning means embodied in the Order Computer Terminal to scan identification code associated with a company and identification codes associated with the products/services desired. This product and company information is stored in the Order Computer Terminal along with credit information retrieved from a plurality of storage means used on credit cards and subsequently transmitted when desired by the consumer to a Central Computer System. The Central Computer System simultaneously receives information from multiple order computer terminals and verifies that the products or services from the desired company are in fact available. The Central Computer System also verifies the credit worthiness of the consumer by searching for such information from credit data bases. When the Central Computer System determines that the desired products/services are available and that the consumer is credit worthy, an order verification signal is sent to the individual consumer's order computer terminal whereupon the consumer verifies that he/she wishes to order the products/services communicated to the central computer system. Once the consumer verifies the order, the automated order and payment system places the order for the products/services desired and provides the appropriate credit reference to the supplier of the product/service. The automated order and payment system capabilities are more fully set forth herein.



B1 4,947,028

1

REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS
INDICATED BELOW.

Matter enclosed in heavy brackets **[]** appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS
BEEN DETERMINED THAT:

Claims 2, 8, 9 and 12 are cancelled.

Claims 1, 3-7, 10-11, 13 and 14 are determined to be patentable as amended.

1. An automated order and payment system which comprises:

[A] a plurality of remote programmable data input/output **[means adapted to optically scan identification code information, and further adapted to accept credit card information obtained from the stored information on credit cards; terminals, and a central data processor, each one of said plurality of remote terminals comprising:**

[A communication means integral to said remote programmable data input/output means;

A memory means integral to such remote programmable data input/output and communication means that allows the storage of computer programs and information derived from printed or transmitted identification code information that has been optically scanned;

A central data processing means with communications capability adapted to receive information from a plurality of remote programmable data input/output means; and

Additional communication means allowing the remote data processing means to communicate with external data bases for credit authorization and product/service ordering purposes]

an optical reader for optically scanning and producing first data representative of visually displayed product or service identification code information;

a payment card reader for inputting payment card information stored on any one of a plurality of user's different payment cards, including means for producing second data representative of such payment card information;

integral memory means for temporarily storing the first data representative of the optically scanned identification code information and the second data representative of the payment card information; and

integral communication means for transmitting the first and second data to said central data processor, subsequent to storage of the first and second data in the integral memory means; and

additional integral memory means for storing computer programs for controlling operation of at least the optical reader, the payment card reader, the integral memory means and the integral communication means;

said central data processor comprising:

2

first central communications means for receiving the transmission of first and second data from the integral communication means of the remote terminals;

order pricing means for calculating an order cost to the user based upon at least the first data received from the remote terminals by the first central communications means;

second central communication means for transmitting the second data and order cost to an external database, and for receiving payment authorization information therefrom;

order confirmation means for providing to the remote terminal, subsequent to receipt of the payment authorization from the external database, a request for the user's confirmation that a product or service represented by the first data should be ordered; and third central communication means for transmitting at least the first data and the payment authorization information to a product/service provider in accordance with the first data, in response to receipt by the central data processor of an order confirmation message from the remote terminal.

3. **[An]** The automated order and payment system according to claim **[2]** wherein **[said]** the payment card reader of at least one remote programmable data input output **[device contains]** terminal comprises a magnetic reader **[to allow input of credit]** for reading information from **[the]** magnetic **[stripe]** stripes on **[credit]** payment cards.

4. **[An]** The automated order and payment system according to claim **[2]** wherein **[said]** the payment card reader of at least one remote programmable data **[input output device contains]** input/output terminal comprises a microchip reader **[to allow input of credit]** for reading information from **[the]** microchips embedded in **[credit]** payment cards.

5. **[An]** The automated order and payment system according to claim **[2]** wherein **[said]** the payment card reader of at least one remote programmable data **[input output device contains]** input/output terminal comprises a laser optical data reader **[to allow input of credit]** for reading information that is optically stored on **[credit]** payment cards.

6. **[An]** The automated order and payment system according to claim **[2]** wherein **[said]** the payment card reader of at least one remote programmable **[input output device allows data entry via a]** data input/output terminal comprises speech processing means **[that allows]** for causing oral instructions to be stored and used as **[digital]** at least one of the first and second data **[input]**.

7. **[An]** The automated order and payment system according to claim **[2]** which comprises a central computer system which **[1 wherein the first central communication means communicates with multiple remote programmable data input/output means]** terminals over a plurality of transmission media.

10. **[An]** The automated order and payment system according to claim **[9]** which contains **[1, the central data processor further including a data base of]** merchant/suppliers **[information concerning availability of products and services from]** which information is extracted concerning the availability **[merchant/suppliers of said products and services]** that a consumer desires to order].

11. **[An]** The automated order and payment system according to claim 10 **[which interacts with a con-**

B1 4,947,028

3

sumer to notify the consumer of the acceptance or], the
central data processor further including means for provid-
ing a message to a remote terminal indicating rejection of
an order, or part of an order, based upon [the ap-
proval/disapproval of credit and/or the] availability/- 5
non-availability of the products or services [desired]
corresponding to the first data.

13. A process for rapidly ordering and paying for
goods and services comprising:

[The optical input by a consumer of a printed or 10
transmitted identification codes used to identify
merchants and goods/services desired by the con-
sumer,

The entry of credit and consumer data by reading
such information from a plurality of storage means 15
resident on credit cards,

The storage and subsequent transmission of such data
to a central computer systems,

The separate processing of such data by the central
computer systems to determine (1) the availability 20
of the product or service desired, and (2) the credit
worthiness of the consumer ordering the products
or services

The notification to the consumer of the approval or
rejection of the sale desired 25

The delivery of the goods or products ordered by the
consumer by prearranged parameters or selectively
designated by the consumer at the time of sale]

a consumer's optically inputting product or service iden-
tification codes used to identify merchants and 30

4

goods/services desired to be purchased by the con-
sumer, into memory of a remote programmable data
input/output terminal,

the consumer's entering of payment data into said mem-
ory by placing a payment card into a payment card
reader integral with the remote terminal,

subsequently transmitting said identification codes and
payment data to a central data processor,

verifying by the central data processor that the product
or services corresponding to the identification codes
are available;

transmitting the payment data from the central data
processor to an external database associated with the
payment data;

the central data processor's receiving payment authoriza-
tion data from the external database;

notifying the consumer of availability of and payment
authorization for, or rejection of, the desired purchase,
and

upon order confirmation by the consumer, delivery of the
goods or services ordered to the consumer by prear-
ranged parameters or selectively designated by the
consumer at the time of sale.

14. [A.] The process for automated ordering of prod-
ucts and/or services according to claim 13 wherein the
[consumer and credit information is] payment data are
derived from a microchip [credit] payment card, mag-
netic storage [credit] payment card and/or optical
data storage [credit] payment card.
* * * * *

35

40

45

50

55

60

65

US005117354A

United States Patent [19]

Long et al.

[11] **Patent Number:** 5,117,354[45] **Date of Patent:** May 26, 1992

[54] **AUTOMATED SYSTEM FOR PRICING AND ORDERING CUSTOM MANUFACTURED PARTS**

[75] **Inventors:** Gary R. Long, Cross Plains; Charles W. Callender, Oregon, both of Wis.

[73] **Assignee:** Carnes Company, Inc., Verona, Wis.

[21] **Appl. No.:** 533,567

[22] **Filed:** Jun. 5, 1990

Related U.S. Application Data

[62] Division of Ser. No. 198,196, May 24, 1988, abandoned.

[51] **Int. Cl.:** G06F 15/21

[52] **U.S. Cl.:** 364/401; 364/408

[58] **Field of Search:** 364/401, 403, 408; 379/96

[56] **References Cited****U.S. PATENT DOCUMENTS**

4,128,758	12/1978	Bukowski et al.	235/433
4,438,326	3/1984	Uchida	235/379
4,466,001	8/1984	Moore et al.	340/825.08
4,468,750	8/1984	Chamoff et al.	364/900
4,503,503	3/1985	Suzuki	364/405
4,511,970	4/1985	Okano et al.	364/401
4,530,067	7/1985	Dorr	364/900
4,578,768	3/1986	Racine	364/560
4,584,648	4/1986	Dlugos	364/464
4,645,238	2/1987	Vincent et al.	283/67
4,674,044	6/1987	Kalmus et al.	364/408
4,695,880	9/1987	Johnson et al.	358/86
4,760,528	7/1988	Levin	364/419
4,799,156	1/1989	Shavit et al.	364/401
4,805,207	2/1989	McNutt et al.	379/89

4,887,208 12/1989 Scheider et al. 364/403
4,899,292 2/1990 Montagna et al. 364/521

FOREIGN PATENT DOCUMENTS

0012068 1/1988 Japan .

OTHER PUBLICATIONS

"Electronic Mail", *Commutation & Transmission*, No. 5, 1982, pp. 21-30.

Acura Integra: Service Manual 1987, Honda Motor Co., Ltd., pp. (1-2).

PTS New Product Announcements, dateline: Grand Rapids, Mich., Aug. 26, 1987, "Quik Quote".

PTS New Product Announcements, dateline: Dearborn, Mich., Mar. 8, 1988, "New Quik Quote Cost Estimating ...".

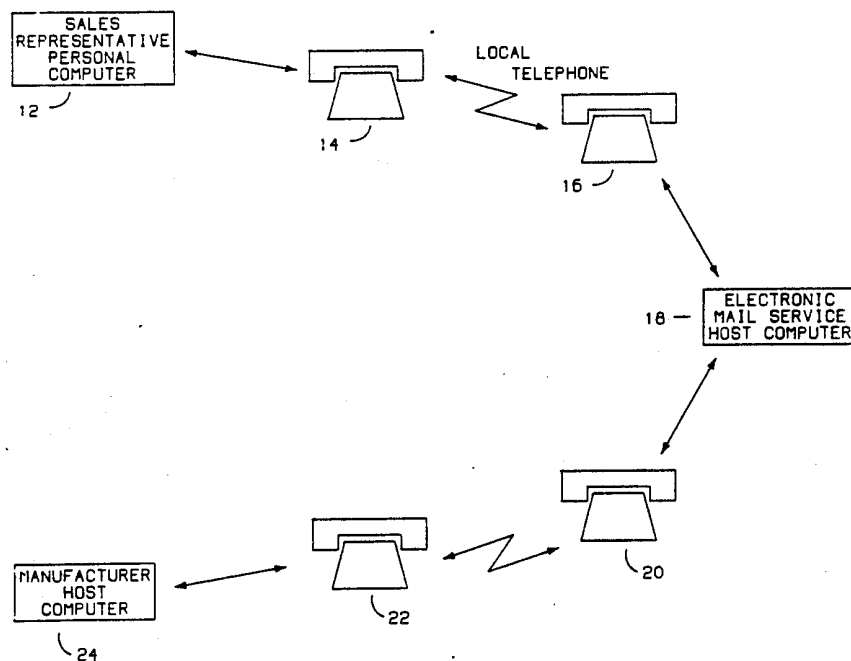
Primary Examiner—Dale M. Shaw

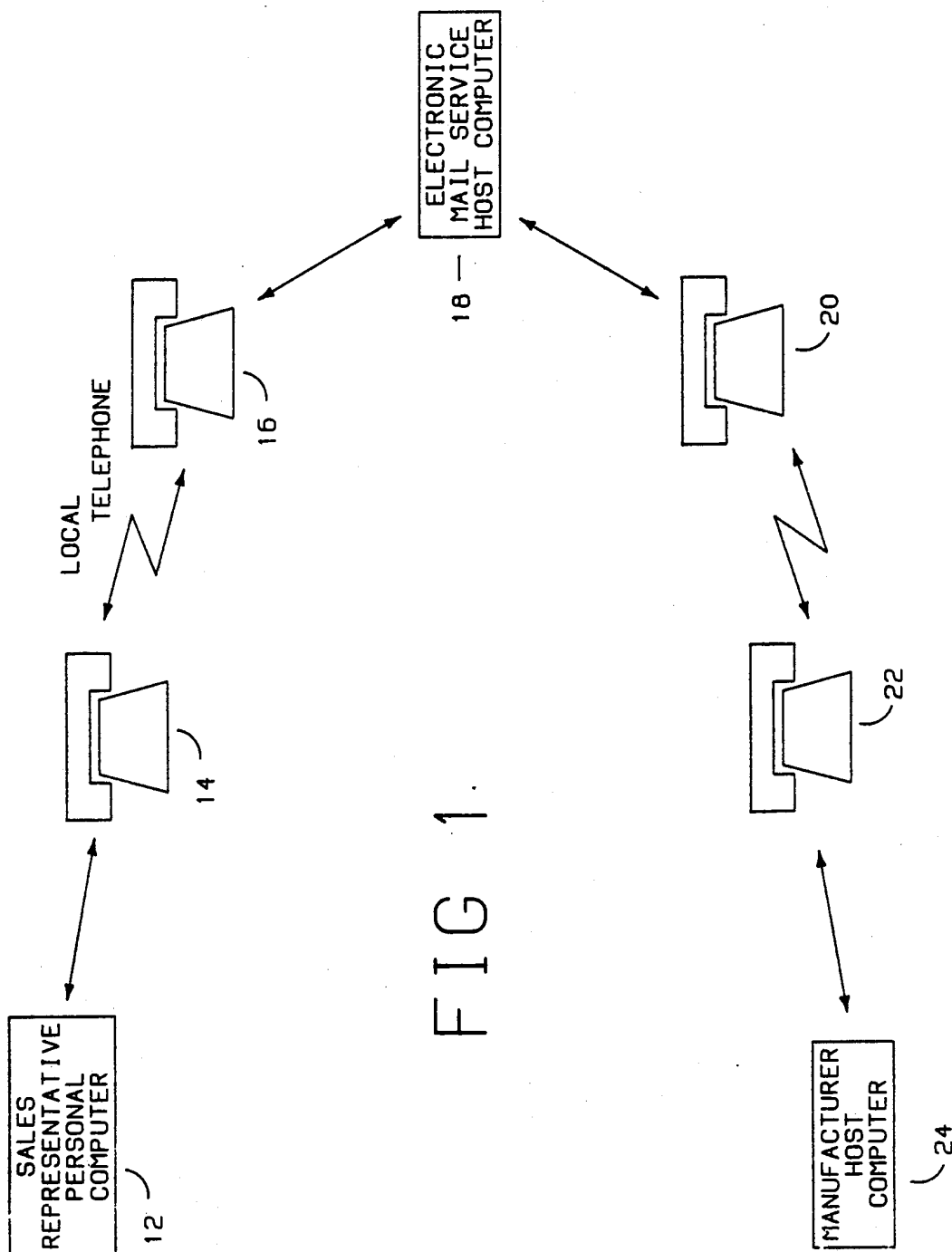
Assistant Examiner—David Huntley

Attorney, Agent, or Firm—Quarles & Brady

[57] **ABSTRACT**

A system is disclosed for the automated pricing and ordering of custom manufactured parts, as for the air handling equipment industry. The system includes software for personal computers of the sales representatives which assists the sales representative in creating product identification codes which specify the specifications of the product to be made. A completed order of such items is deposited in an electronic mail system addressed to the manufacturer. A host computer at the manufacturer periodically polls the electronic mail system for communications and then either prices the quote or processes the order.

7 Claims, 7 Drawing Sheets



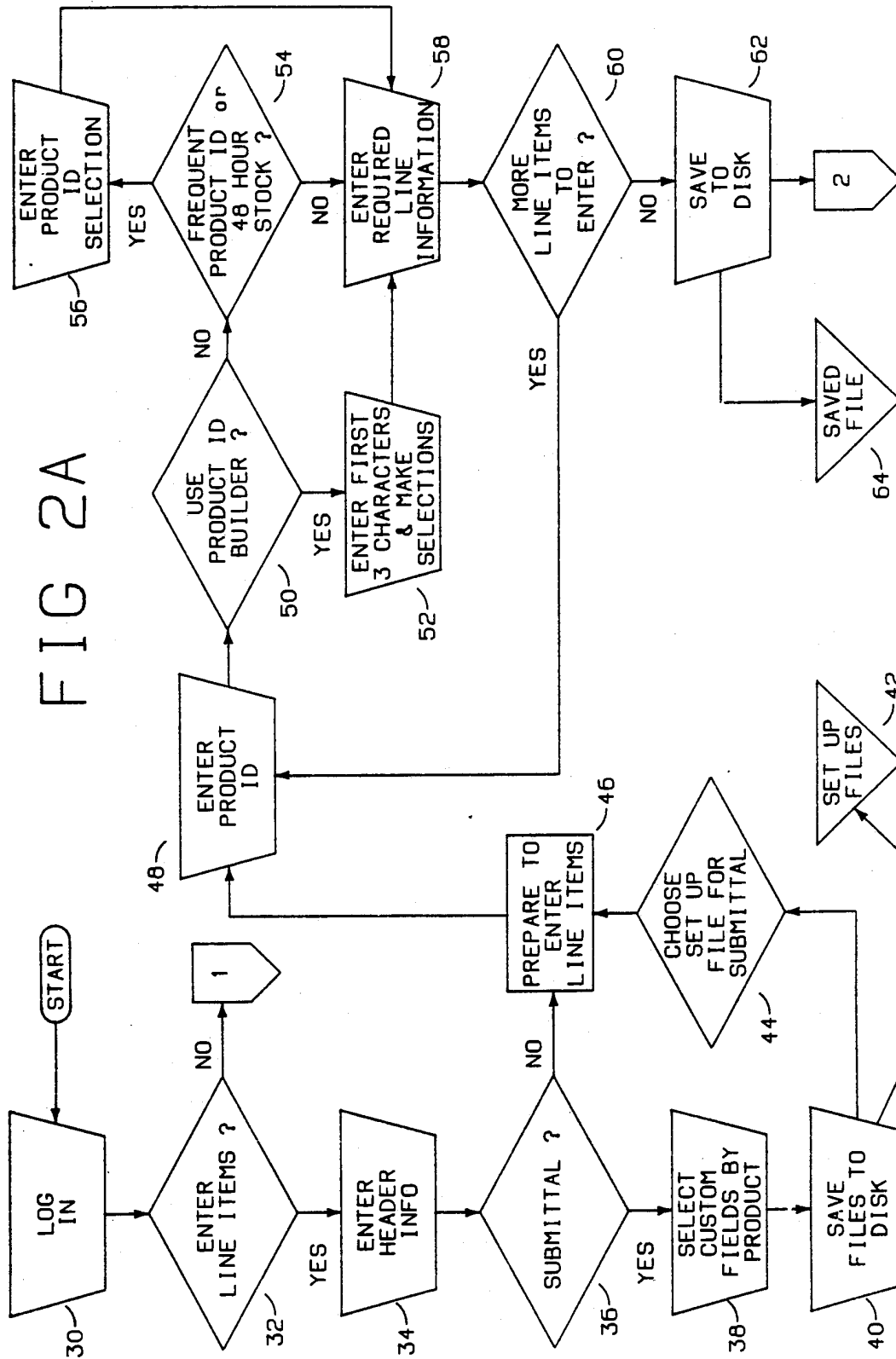
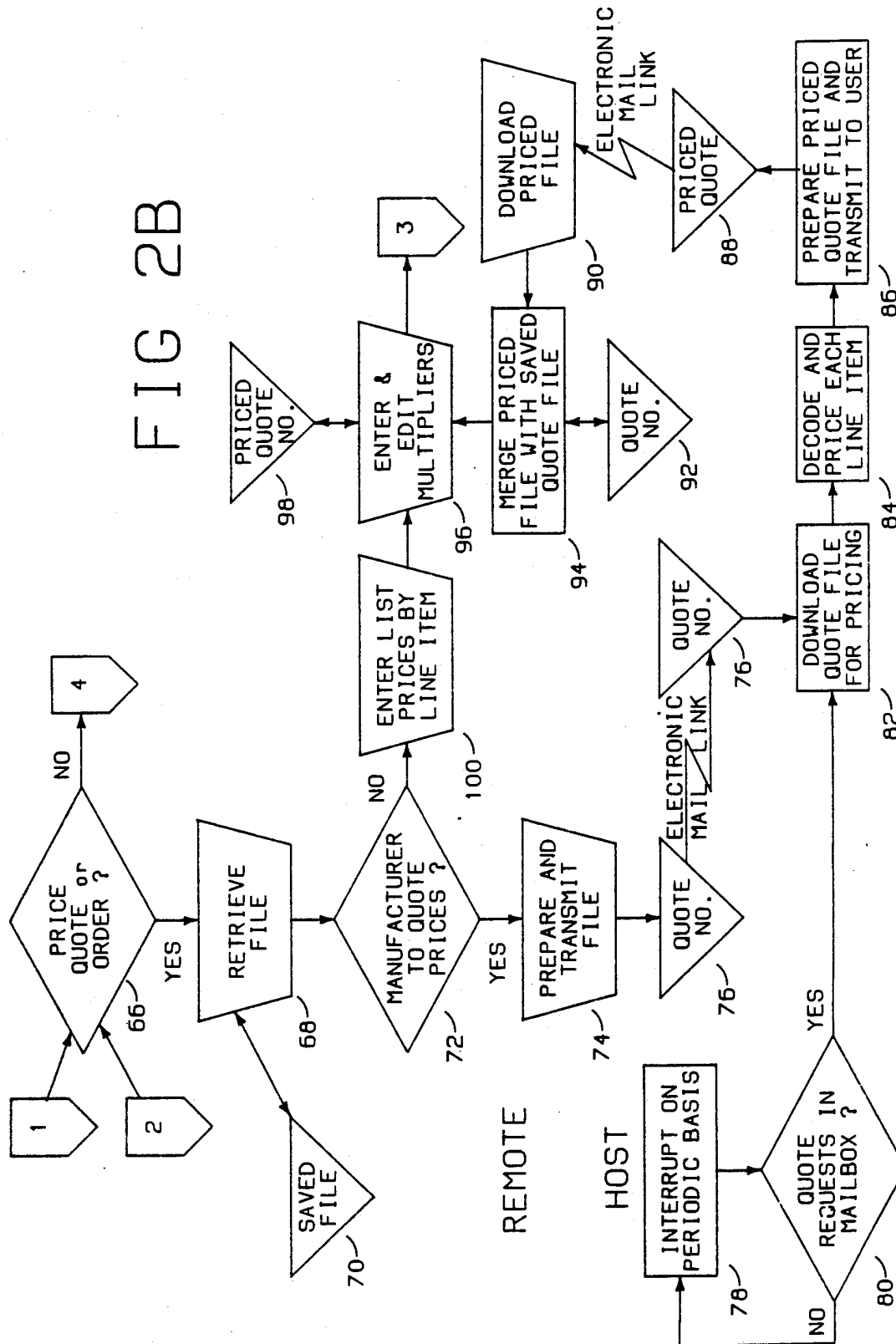


FIG 2B





MAIN MENU

1	-BEGIN NEW JOB/QUOTE	}	GOTO LINE ITEMS
2	-SKIP HEADER		
3	-CREATE A SUBMITTAL		
4	-PRICE/PRINT SPREADSHEET		
5	-PRINT SUBMITTAL		
6	-TRANSMIT FOR PRICING		
7	-RECEIVE PRICING		
8	-SETUP		
9	-NEW JOB		
0	-QUIT		0 -ORDER

U -UTILITIES

FIG 3

QUOTATION SYSTEM

REPRESENTATIVE NUMBER --M99

DATE : 11/02/87

PRODUCT ID: VUBB12L4111N20C1

QUANTITY: 1

TAG:

COMMENT:

FAN RPM:

Current line: 20	Lines saved: 20	Use (ENTER), or to move cursor
F1--EXIT/MAIN MENU	F5--COMMON PI LIST	F10--HELP MENU
F2--EXPLODE PID	F7--48 HR STOCK LIST	PgUp--SHOW PREV LINE
F3--PID MENU -ON	F8--DELETE THIS LINE	PgDn--SHOW NEXT LINE

FIG 4

U.S. Patent

May 26, 1992

Sheet 7 of 7

5,117,354

QUOTATION SYSTEM

REPRESENTATIVE NUMBER --M99

DATE : 11/02/87

PRODUCT ID: VUBB12L4111N20C1 UPBLAST CENTRIFUGAL ROOF VENT

DESIGN	B SERIES	MOTOR	1 OPEN DRIP PROOF
SIZE	12	LISTING	N NON-LISTED
H.P.	L 1/4 HP	FINISH	20 MILL FINISH
RANGE	4	WHEEL	STD
ELECT'CL	11 115V-SINGLE	SCREEN	NO SCREEN
		CAP	C STD CURB CAP
		OPTIONS	1 DISCONNECT

F1--EXIT W/O SAVING THIS PID USE ... HOME or END to MOVE CURSOR
 F2--SAVE THIS PID USE SPACEBAR TO DISPLAY OPTIONS

FIG 5

5,117,354

1

AUTOMATED SYSTEM FOR PRICING AND ORDERING CUSTOM MANUFACTURED PARTS

This is a division of application Ser. No. 07/198,196 filed May 24, 1988 now abandoned.

The present invention relates to systems for pricing and ordering goods in general, and relates, in particular, to an automated system for implementation by a manufacturer having numerous widely separated sales representatives organized so that the sales representatives can obtain pricing information, and place orders for the goods to be manufactured, on an efficient and automated basis from the manufacturer.

BACKGROUND OF THE INVENTION

In some industries, such as the business of manufacturing air distribution equipment for commercial and industrial applications, many if not most of the goods are manufactured specially in custom fashion for each particular order. Such manufacturing on a custom basis is needed, and desirable, because each of the pieces of equipment in such an industry must be made in an inordinately large number of sizes. For example, steel air duct registers may be available in any quarter inch increment of length or width size over a very large size range. The permutations of mathematically available sizes in which such registers might be ordered makes it obvious that either an enormously large inventory must be maintained or individual registers must be manufactured in particular sizes for particular jobs on a custom basis, as needed. It is this practice of the custom manufacture of air handling equipment that has become common in the industry.

One difficulty in administering and maintaining such a custom manufacturing business is the problem of constructing and pricing detailed orders for particular jobs. As in many industries, in the air distribution equipment industry, sales are actually made in the field by independent sales representatives. Such sales representatives are typically not employees of the manufacturer, but are manufacturer's representatives who may represent other complementary companies as well. It can become an extremely complex for such sales representatives to learn the process of specifying, pricing, and ordering items in a custom manufacturing industry in which there may be an overwhelmingly large number of items which can be ordered. Previously this industry has adopted a practice in which manufacturers send to each of their sales representatives a large book which has prices fixed for some period of time, i.e., a sales year, and contains in detail the part number, and size information for each part number, together with the price for each sized part. The use of such books is satisfactory for such an operation, but can lead to difficulties with regard to access to the various part numbers in question and errors introduced by the sales representative needing to gain pricing information for large numbers of parts. Since for each part there are codes for various sizes, styles and options, clerical errors become easy to occur.

Such a system has an additional level of complexity brought about by the fact that the sales representatives receive a discount off of list price, with the discount potentially varying by both the size of the order and the numbers of a particular item which are ordered. This also leads to possibilities of mistake and error in the ordering and pricing process.

2

The entry of orders electronically into electronic data processing apparatus for pricing and totalling sales of items is generally known in the prior art. The prior art contains examples of systems, for example, in which the number and identity of items to be sold is placed on cards which are electronically read so that total price and order information can be calculated. Such systems are usually local in their operation, however, allowing pricing and ordering information requests only to be implemented by persons in a restricted locale or requiring the actual physical transportation of media for processing to the data processing facility.

It is also generally known in the art that remote terminals can be utilized for financial transactions. There are many examples known in the prior art, such as for example, automatic teller machines, in which remote transactions are conducted by a customer with the data regarding the transaction being transmitted by electronic communication, usually over a leased or dedicated telephone line, to a remote host which receives the information, processes the request, and communicates acceptance and verification of the exchange back to the remote terminal. The majority of such systems are implemented on a basis which requires an essentially full-time communication link between the host and the remote terminals, so that immediate access to the host is always available to the terminal. In a system in which the usage at each individual remote terminal is relatively infrequent, the use of such dedicated telecommunication links suffers from a disadvantage in that the maintenance and carrying costs of such links can be out of proportion to the economic value of the communication service provided.

SUMMARY OF THE INVENTION

The present invention is summarized in that a system for the remote pricing and entry of orders for custom manufactured items includes a central data processing facility at the manufacturer connected to a telecommunication link to an electronic mail service host, a remote station for a sales representative including a personal computer and a remote telecommunications link, and an electronic mail service host having telecommunication links to both the sales representative personal computer and to the manufacturing host and organized to have electronic "mailboxes" for messages addressed to either of the sales representative or the host, the manufacturer's host including means for polling the mailbox assigned to the host on a periodic basis so that messages transmitted by the remote sales representatives are processed in an expeditious fashion.

It is an object of the present invention that a method is provided for the obtaining of pricing information and the entering of orders for custom manufactured equipment which is simple, quick and efficient in its operation, and is automated, without requiring dedicated telecommunication links to each sales representative in the field.

It is another object of the present invention to provide a method of processing pricing and ordering transactions in which sales representatives in the field deposits pricing or ordering requests with an electronic mail facility host which is periodically polled by the host computer of the manufacturer so that orders receive relatively prompt attention without requiring dedicated telecommunication links.

It is an object of the present invention to provide a system for efficiently pricing and ordering the manufac-

5,117,354

3

turer of air distribution equipment in a prompt and expeditious manner while lowering the probability of erroneous price quotes and incorrect errors.

Other objects, advantages, and features will become apparent from the following specification when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic illustration of the linkage between the parts of a system constructed in accordance with the present invention.

FIGS. 2A through 2C are flow chart diagrams illustrating the method of operation of a system constructed in accordance with FIG. 1.

FIG. 3 is an illustration of a main menu of the system of FIGS. 1 and 2.

FIG. 4 is an illustration of a line item entry screen of the system of FIGS. 1 and 2.

FIG. 5 is an illustration of an exploded product ID screen of the system of FIGS. 1 and 2.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Illustrated in FIG. 1 is a schematic illustration of how the custom manufacturing pricing and order entry system in accordance with the present invention is arranged. A variety of sales representatives in the field are each equipped with a personal computer 12 having processing capabilities, local memory and long term storage such as disk drives. Those personal computers 12 may be installed at the office locations of the sales representatives or may be portable units which they may carry with them to their home or other remote locations. At any remote location in which there is a telephone, the personal computer 12 can be connected to a local telephone 14 through a data communication device, typically a modem, so that digital information can be transmitted through the local telephone lines. Through such a local telephone linkage, the local telephone 14 can call a local telephone node 16 of an electronic mail system. The electronic mail system consists of an electronic mail service host 18, which is typically a large mini or main frame computer located in a remote facility which is connected by leased or dedicated data communication lines with local telephone nodes spread throughout the United States. Such an electronic mail service will have a local node 16 at or near the location of each sales representative and will also have a local node 20 at or near the location of the manufacturer. Several competing electronic mail services, including an electronic mail service host, such as 18, and appropriate local nodes spread throughout the country, are available. One such service is the EASYLINK (TM) service offered by Western Union. Some systems may use a digital interconnect carrier, such as TYMNET or TELENET, to communicate with the various nodes around the service area. From the local telephone node 20, the data communication link extends over local telephone lines to a telephone access 22 connected to the manufacturer's host computer 24.

In an electronic mail system, such as that implemented by the electronic mail service host 18, communication need not be instantaneous between remote devices which have access to the electronic mail host 18. Typically the overhead software and procedures are implemented in an electronic mail service system assign a series of address location, or "mailboxes," which have an address or access code associated with each of the

4

users of the electronic mail system. Thus each sales representative having a personal computer 12 will have assigned to it one or more mailboxes in the system of the mail service host 18 and the manufacturer host computer 24 also has assigned to it one or more mailboxes within the electronic mail service host 18. A preferred method of implementation is for the manufacturer host to have assigned to it at least two electronic mailboxes, one assigned to receive requests for quotations and one assigned to receive orders. During any given access to the electronic mail service, any remote device associated with it, such as one of the personal computers 12, or the manufacturer host computer 24, can either deposit files onto the electronic mail service host 18, to be stored in the mailbox of the addressee, or can read items out of its own mailbox to "download" them into the computer which is accessing the electronic mail service host 18.

Thus it is envisioned within the system of the present invention that, in the procedure described in more detail below, the sales representatives utilize their personal computers 12 to construct quotations or orders of items and part numbers or product IDs. Those listings of items and product IDs are deposited by the sales representative's personal computer 12 onto the electronic mail service host 18 by use of the linkage provided by the telephone access provided by telephone nodes 14 and 16. The data file, i.e. quotation or order, consisting of the items and product IDs are stored by the electronic mail service host 18 in a mailbox associated with the manufacturer host computer. Periodically on a fixed time period basis, such as once every twenty minutes, the manufacturer host computer 24 accesses the mail service host 18 through its telephone connections 20 and 22 to interrogate its mailbox to determine if any messages have been left for it. The manufacturer host computer 24 then downloads the file containing the items and product IDs together with information identifying the sales representative who has submitted the request. The host computer 24 can then calculate out pricing information based on the items and product IDs. The resulting information in the form of a priced quote is then uploaded from the manufacturer host computer 24 through the telephone linkage 20 and 22 into the electronic mail service host 18, where it is stored in the mailbox designated for the particular sales representative who had previously placed the pricing order request. Thus, if the sales representative again uses his personal computer 12 to examine the contents of his mailbox after some reasonable wait after he has deposited his request for pricing, such as for example, an hour later, the sales representative will receive a priced quote file which can be downloaded from the electronic mail host 18 onto his personal computer 12 in such a way that the file contains not only the items and product IDs which the sales representative entered into the system, but now also contains complete pricing, and discount information as supplied directly from the manufacturer. In this way, the sales representative can prepare and implement pricing quotations and proposals in a much more accurate and efficient manner, and with a much closer linkage to the actual pricing and parts information supplied directly from the manufacturer.

Illustrated in FIGS. 2A through 2C, together with FIGS. 3-5, is an illustrative schematic flow chart of the manner of operating the computer program or software in the sales representative personal computer 12 and the manufacturer host computer 24 necessary to implement

5,117,354

5

the system as described in general above. The system is designed to allow flexibility to the sales representative either to do local pricing, using traditional printed reference materials, or to do automated pricing by submitting pricing requests to the manufacturer through the electronic mail service. The system is also capable of not just requesting pricing information, but also actually placing orders with the manufacturer for the construction and shipment of custom goods to addresses as indicated by the sales representative.

Illustrated beginning in FIG. 2A is the flow diagram for the operation of the computer software operating in a method in accordance with the present invention. The software begins, at step 30 with a log-in procedure in which a user, i.e. a sales representative, is interrogated as to his password to ensure that he has proper authority to access the system. Such system log-ins are well known to the prior art. The next step, illustrated at step number 32, is a decision step to ascertain whether the user desires to enter line items i.e. to define items of goods to be priced or ordered. This decision step 32, is implemented by the user accessing or choosing items from a menu. Illustrated in FIG. 3 is a sample of such a menu which can be used as the main menu for a system implemented in accordance with the present invention. The menu as illustrated in FIG. 3 gives the user the ability to begin new jobs or orders, to create a submittal, to expand items to a price/print spread sheet, to print a submittal, to transmit an item for pricing, or to receive pricing. The menu also allows the user to set up new jobs, to quit, to order the manufacture and shipment of goods, or to select utilities which may be utilized to update his local software. Note that several of the selections on the menu lead the user into the preparation and editing of particular line items. Those selections would be represented in the flow chart of FIG. 2A as a yes to the question presented at step 32. If the user does not choose to enter line items, the flow chart branches to FIG. 2B. If the user does desire to enter line items, such as for example selecting one of the first three selections from the main menu illustrated in FIG. 3, the program then proceeds to program step 34 where the user is asked to enter appropriate header information for the quote or order. This information can be simply requested by screen prompts which require information to be entered about the identity and name of the ultimate customer, and any job numbers or quotation numbers associated with the quotation or order being prepared. The header information may also include a note pad area for the user to write down notes about the job and to enter the effective pricing date requested. This is a free form location in which miscellaneous information not elsewhere appropriate can be entered for later transmission to the manufacturer.

The next item on the flow chart of FIG. 2A is a decision step inquiring whether or not what is to be prepared is a submittal, and this step is illustrated at 36 in FIG. 2A. A submittal is a document not for use for communication between the sales representative and the manufacturer, but is a document intended for submission to the ultimate customer, i.e., in the business of air distribution equipment, to the engineer or construction company which will be using the products. Therefore, while the system for communicating part identification and ordering information between the manufacturer and the sales representative is within the control of the manufacturer, the system must also have the ability to print submittals in a wide variety of styles and for-

6

mats as might be required by particular customers. Thus, if a submittal is to be prepared, the system then allows the sales representative to select and prepare custom fields by product type and to generally have editorial control over the style of format and appearance of the submittal form. This process is reflected by step number 38. Because many sales representatives will deal repetitively with the same customers, and because the creation of a submittal form for a particular customer may require a significant amount of time and effort, the system then allows, at program step 40, for the sales representative to save a file on disk storage with the file consisting of a set up file containing information on the custom fields necessary for a submittal document prepared in accordance with the wishes of the particular customer. The file saved on disk is represented at 42 in FIG. 2A. In any event, the user then proceeds to a decision step 44 in which the user chooses the set up file to be used for the particular submittal to be prepared. This set up file may be one which is custom prepared at step 38 or may be one which is retrieved from the set up files 42 at step 40. In any event, this submittal form is saved with the line items entered subsequently for use in preparing the ultimate submittal to be prepared for the customer for the goods. Thus, regardless of whether a submittal is to be prepared or not, ultimately the user proceeds through the system to prepare to enter line items as indicated by the program step 46.

Illustrated in FIG. 4 is the line item entry screen presented on the personal computer of the user for the entry of line items representing individual items to be manufactured for the particular quotation or order being created by the sales representative. The line item entry screen has as its first line the entry of the product identification code or product ID. This is represented at program step 48 in FIG. 2A. Note, as illustrated in FIG. 4, that the product identification code is a lengthy alpha-numeric character string. In a business such as the manufacture of equipment for air distribution, and other industries in which a large number of custom sized fabricated parts are to be constructed on a custom basis, an extremely large product ID is useful to encode sufficient specifications about the particular part, style, color, finish, and, in particular, size. Therefore the product identification may typically have a large number of digits, varying from twelve to as many as twenty which may be numeric or alpha-numeric in character. This product identification code thus contains more information than just the style of product but contains imbedded codes indicating the actual size and finish of the goods to be manufactured. It is the complexity of these product identification codes and their proper usage and implementation that makes an automated system constructed in accordance with the present invention particularly useful for avoiding clerical errors in the entry and pricing of products specified in this manner.

The next step in the program, as illustrated in FIG. 2A, is represented as a decision step 50 asking whether the user desires to use the product ID builder. The product identification code builder is represented by an option, available by function key as illustrated in the line item entry screen of FIG. 4. Note that function key F2 is labeled "explode PID." Using the product ID builder, by pressing function key F2 from the line item entry screen of FIG. 4, makes available to the user the exploded product identification screen illustrated in FIG. 5 which is the product ID builder. This screen

5,117,354

7

provides an input for the product identification code so that the software may refer to tables contained in the software to determine from that product identification what the size, kind and nature of the goods are. Thus the screen has a series of parameters, illustrated in FIG. 5 with reference to a roof vent, appropriate for the kind of goods indicated by the product ID. All of the selections contained in the various parameters listed in the screen of FIG. 5 are all contained within information embedded in the product identification code, or product ID, shown at the upper left corner of that screen. The screen embodies the method by which the user accesses all the information which is embedded in that code so that the sales representative can be sure that all of the appropriate options which are desired for a particular item are properly requested by properly encoding the product ID. This is illustrated at program step 52 in FIG. 2A.

In the detailed operation of the product identification code builder as illustrated by FIG. 5, the user first enters the first three characters of the product ID in the upper left hand corner of the screen of FIG. 5. The computer program operating in the personal computer 12 of the sales representative code with a representative product ID code and then fills out the remainder of the screen with the product specifications for the item represented by the representative product ID code. Thus the representative product ID code functions as a default product ID which is completed by the software. The sales representative can then proceed through the various option items on the screen, by moving the cursor around the screen, and make changes to the displayed product specifications for the default product displayed. For example, if the sales representative desires to change the size, indicated as "12" in the example of FIG. 5 to, for example, "14," the sales representative moves the cursor down to the line indicated by size and replaces the number "12" with the number "14." The system software then automatically updates the completed default product identification code by changing the embedded code 12 to 14, indicating the size within the product identification code itself. Thus the sales representative can move through the options available on the exploded screen of FIG. 5 and change the product specification or option items while the system automatically updates the code of the product ID to include the changes so that the user does not have to learn each of the codes for the specifications of each product which is available. It is this process of selecting option items which are updated into the product identification code which is represented by program step 52 in FIG. 2A in which the first three code characters are entered and then the various item selections for specifications and options are made. When the sales representative has completed selection of all his options in the screen in FIG. 5, the sales representative may choose function key F2 to save this product identification code and return to the product ID selection screen of FIG. 4 for a new line item entry and to indicate to the software that the updated product identification code is complete.

If the product ID builder is not used, the program branches to decision step 54 which asks if the desired item is a frequent product ID or requested from forty-eight hour stock. If it is a frequent product ID with which the sales representative is familiar, the sales representative need not call the PID menu as shown by FIG. 5, but can enter product ID selections directly as

8

indicated by step 56 in FIG. 2A. Similarly, a manufacturer may maintain a list of standard items available on forty-eight hour stock, which have product ID selections listed for them so that the sales representative can merely refer to that list to gain the appropriate product ID code for selection. In any event, regardless of the manner in which the product ID is constructed, the product ID is then used at step 58 in conjunction with the other required information, such as quantity, to create a line item to include in the job for quotation or ordering. As shown in FIG. 2A, the program then proceeds to step 60 which is a decision step requesting whether there are more line items to enter. If there are more line items to enter, then the program returns to step 48 where the user can again either request the product ID builder or can enter product IDs directly. If the user has completed a particular job with no more line items to enter, the program proceeds to step 62 in which the program saves to disk a file, indicated at 64, which now contains a job number plus a list of items and associated product ID codes representing a total job to be either priced or ordered on behalf of a particular customer. In the continuing flow chart illustrating the operation of the present system, continued in FIG. 2B, the next step is a decision step indicated at 66 which asks whether it is desired to price the quote or order. If a pricing step is not to be performed, the program proceeds to the steps illustrated in FIG. 2C. If a pricing procedure is to be performed, the program then proceeds to program step 68 in which the file containing the list of items to be priced is retrieved. The saved file which is retrieved is indicated at 70. This saved file can either be the same save file which was just saved to disk at step 64 in FIG. 2A, or can be any other saved file currently carried on the system. The next decision step in the flow chart is indicated at program step 72 and represents alternative methods for quoting prices for the items in the job. The sales representative at the remote location with his personal computer can either manually quote the item prices himself or can transmit the quote to the manufacturer for pricing. Assuming that the sales representative desires to transmit the quote to the manufacturer for pricing, he proceeds to program step 74 in which the quote file is prepared for transmission and to program step 76 in which the actual quote file, referenced by quote number, is transmitted to the manufacturer. This file, referred to by quote number, includes the quantity and product identification code for each item in the job. As illustrated schematically in FIG. 2B, the quote is transmitted by electronic mail linkage from the remote location, or the personal computer of the sales representative, through the electronic mail system linkage to the manufacturer host computer. This linkage is the one illustrated in FIG. 1.

The manufacturer host computer, 24 in FIG. 1, is available for other processing on an on-going basis. Periodically it has an interrupt, indicated by program step 78 in FIG. 2B. This interrupt is connected to initiate a procedure of inquiry into the electronic mail system on a periodic basis. The period should be such so that the host system is reasonably responsive to inquiries from sales representatives. Such a periodic basis might be once every fifteen or twenty minutes. When such an inquiry is made, the program then proceeds to program step 80 in which it is inquired if there are any quote requests in the mailbox for the manufacturer host. Imbedded in the step is the concept that the host would automatically have telecommunication parameters built

into it so that it could dial the access number of the local node of the electronic mail service system, and be connected therethrough to the electronic mail service host, and thereafter present to the electronic mail service host the appropriate mailbox address which has been assigned to the manufacturer. Such parameters can be built into the telecommunication software embedded in step 80 so that the system automatically polls the electronic mail system and looks for communications placed in its mailbox. If such communication is in its mailbox, then the program proceeds to download the same quote number 76, which has been transmitted over the electronic mail system, through a download step indicated at 82 into the manufacturer host computer. The manufacturer host can then decode each item on the quote and price each line item, as indicated at program step 84. That manufacturer's host then reassembles the file as a priced quote for transmittal to the user and transmits it to the user at program step 86 as a priced quote indicated at 88. Again the quote is transmitted into the electronic mail service system, this time addressed to the mailbox associated with the personal computer of the sales representative who transmitted the request for quote.

At the remote location, the sales representative uses his personal computer to download the priced file at program step 90. Under normal operation, the sales representative would not have to be in personal communication with the manufacturer to know that a file is available for downloading. The sales representative would merely up load the quote request into the electronic mail system and then return at some reasonable later time, as for example an hour later, and poll his electronic mail mailbox for the priced quote. Since the host system is monitoring and responding on a periodic basis, typically something like twenty minutes, unless there has been a defect or problem with the system, after such a reasonable wait the sales representative will find his priced quote in his electronic mail mailbox ready for downloading at program step 90. The software of the remote location then merges the priced quote file with the saved quote file indicated at 92 at program step 94. The program then proceeds to enter and edit the appropriate multipliers for the sales representative at program step 96. This procedure creates a completed priced quote, stored by number, which is stored to disk as indicated at 98. At step number 72, if it was decided that the manufacturer was not to quote the prices, then the sales representative need to refer to the manuals and other printed reference material to determine the prices for each line item and then the sales representative, at program step 100 would then manually enter the list prices by line item into the system. From step 100 the program would again proceed to step 96 to enter multipliers and commissions to complete the fabrication of the price quote number. The program then proceeds to the steps illustrated on FIGS. 2C.

On FIG. 2C, the first step illustrated is a decision step 102 as to whether the user desires to send an order to the manufacturer. The program reaches step 102 either from step 96 in FIG. 2B or from step 66 in FIG. 2B. If the user does desire to send an order to the manufacturer, the program branches to step 104 in which complete order transmittal information is obtained from the user. The order transmittal information required includes invoice name and address, an order number, a customer order number, a ship to address and other information such as shipping date, telephone numbers

for inquiries and special shipping instructions. Before the order is transmitted the program proceeds to program step 106 in which job information questionnaire (JIQ) entry information is requested from the user. This information includes information about the project for which the materials are being ordered, such as the project name, address, the name and address of the general and subcontractors, name and address of bonding agencies and name and address of the project owner. Because such JIQ information is required by the manufacturer to properly fill such an order, the program will refuse to advance to order transmittal until the requested JIQ information is supplied.

Once the appropriate information has been assembled, at program step 108 the priced quote is retrieved from disk, indicated at 110 and is transmitted to the manufacturer as an order with the completed order transmittal and JIQ information via the electronic mail link.

At the manufacturer, the manufacturer host computer, during its periodic polling of requests and other items placed in its mailbox, senses that an order has been placed in its electronic mailbox and downloads the order at program step 112. Then the manufacturer host proceeds to verify the accuracy of the order at program step 114 and, assuming that it is accurate, prints the order for scheduling and credit approval at 116 resulting in an order number 118 in hard copy. A copy of the hard copy printout from step 118 is sent to the sales representative to confirm receipt and processing of the order. This is the end, indicated at 120, of the ordering process. The goods are then manufactured and shipped in accordance with the order instructions. Order confirmation can be retransmitted back to the sales representative, as desired, again using the electronic mail linkage.

Another option available within the system available to the sales representative on his personal computer 12 is to print what is called a submittal report. This option is indicated by the branch step indicated at program step 124 in FIG. 2C. A submittal report is a form for submission to the customer in the customer's own desired format and style that is, in essence, a bid on a particular job or project. If a submittal report is to be prepared, the program proceeds to program step 126 in which a set up file and a quote are loaded in and assembled. The set up file, indicated at 128 is the same type set up file referred to at 42 in FIG. 2A which consists of the print parameters and formatting parameters for a submission report in the style desired by a particular customer. The quote number, indicated at 130 has a price quote previously saved disk. Once the quote has been processed in the style and fashion desired and appropriate for the set up file, the system then proceeds to program step 128 in which the print settings are established and then to program step 130 in which the custom report based on the print settings in the set up file is printed thereby producing a printed submittal report 132. This is the report which can then be sent to the customer. As indicated at 134 this is the end of this routine.

The last option available on the system is maintenance of the parameters for the system maintained at the personal computer 12. This is indicated by system maintenance request indicated at program step 136. If no system maintenance is to be performed, the program ends at 138.

5,117,354

11

The first option available if system maintenance is to be performed is to create set up files indicated by program step 140. These are the files used to set parameters for printing and formatting of the submittal reports to be prepared for particular customers. If this routine is to be implemented, the program then proceeds to step 142 in which the user sets parameters by product group and identifies the type and format of the documents to be prepared. Once the set up parameters are determined, at program step 144 these parameters are saved to disk to create a set up file 146.

Another system maintenance option available is to edit and create product IDs indicated at program step 148. If this option is to be utilized the program then proceeds to a program step 150 in which common product IDs, or PIDs, are added, changed or edited as appropriate and the results are stored in a common PID file indicated at 152.

There is also an option, indicated at program step 154 to perform general system maintenance which includes, as indicated at program step 156 with adding or deleting file on the system. At the end of the performance of system maintenance, the program ends at 158.

Thus, in the system of the present invention, a system and a method for utilizing it is enabled for the accurate pricing of custom manufactured goods of a complex and detailed nature. The system is made accurate and efficient by providing an electronic mail linkage between the remote sales representative and the manufacturer. The sales representative prepares his quote by entering the various line items and identifying the part numbers and then assembles the quote and transmits it to the electronic mailbox associated with the manufacturer. The manufacturer's host computer periodically polls the electronic mailbox assigned to it to look for requests for quotations or orders. If a request for quote is there, the host downloads the request, prices it, and transmits it back to the electronic mailbox associated with the requester. If an order is present, it is received and processed. Since this is done on a periodic basis, the sales representative who has requested a quote knows, with some reliability, that he can return and download his priced quote after a predetermined wait period. Since the sales representative is utilizing the manufacturers system for the pricing, he runs less risk of making erroneous pricing decisions by making clerical errors in referring to printed reference materials. In addition, since the actual order can be transmitted electronically through the same electronic mail linkage, the transmittal and processing of orders proceeds much more expeditiously and orders are received by the manufacturer within the hour of transmittal from the sales representative in the field.

It is to be understood that the present invention is not limited to the particular construction and arrangement of parts illustrated herein, but embraces all such modified forms thereof as come within the scope of the following claims.

We claim:

1. A method utilizing a computer with memory 60 means and display means to prepare product identification codes for custom manufactured parts for which information and specification, including physical description, for the parts are encoded in the product identification, the method comprising the steps of

(a) displaying a product ID display to a user on the display means, the product ID display including a location for the product identification code and a

12

plurality of display locations for product specifications which are encoded by the product identification code;

- (b) receiving from the user a plurality of characters representing the beginning of a product identification code;
- (c) displaying on the display means a completed default product identification code including the received characters and also displaying the product specifications, including physical description, for the completed default product identification code;
- (d) receiving from the user changes to the product specifications displayed for the completed default product identification code;
- (e) the computer changing the completed default product identification to an updated product identification code to incorporate any changes to the product specifications received from the user by incorporation of those changes into the displayed product identification code; and
- (f) receiving from the user an indication that the updated product identification code is complete so that a complete product identification code for the desired product specifications is created without the user needing to understand the manner of encoding of the product specifications in the product identification code.

2. A method as claimed in claim 1 wherein the displaying of step (a) is a video display on a CRT screen of a personal computer.

3. A method as claimed in claim 1 wherein the receiving of step (b) includes receiving the first three characters of the product identification code.

4. A method as claimed in claim 3 wherein the displaying of step (c) includes displaying at the location on the product ID display a completed default product identification code corresponding to a standard part specified by the first three characters received.

5. A method as claimed in claim 1 wherein the receiving of step (d) is accomplished by the user replacing default product specifications on the display means with the desired specifications.

6. A method as claimed in claim 1 wherein the custom manufactured parts are parts for air distribution equipment.

7. A method for pricing custom manufactured parts utilizing a remote personal computer operated by a sales representative, a manufacturer host computer operated by a manufacturer and an electronic mail system in which messages can be left in electronic mailboxes assigned to particular users, the method comprising the steps of:

- (a) preparing on the personal computer of the sales representative a quotation including a list of items by product identification code, the product identification code for each item including imbedded information as to the physical specifications of the item to be manufactured, the product identification code for each item being developed through a method comprising the steps of

- (1) displaying for the sales representative on the personal computer a product ID display including a location for the product identification code and a plurality of product specifications which may be encoded by the product identification code,
- (2) receiving from the sales representative a plurality of characters representing the beginning of a product identification code,

5,117,354

13

- (3) displaying on the personal computer a completed default product identification code including the received characters and also displaying the product specifications for the completed default product identification code,
- (4) receiving from the sales representative changes to the product specifications displayed for the default product identification code,
- (5) the personal computer changing the product identification code displayed to the sales representative to match the changes received to the product specifications, and
- (6) receiving from sales representative an indication that the updated product identification code is complete;
- (b) transmitting the quotation including the list of items by product identification codes into the elec-

14

- tronic mail system addressed to the mailbox of the manufacturer;
- (c) periodically polling the mailbox of the manufacturer by the manufacturer host computer;
- (d) downloading the quotation from the mailbox of the manufacturer into the manufacturer host computer;
- (e) in the manufacturer host computer, pricing each item in the quotation;
- (f) transmitting the quotation with the pricing for each item from the manufacturer host computer into the electronic mail system addressed to the mailbox of the sales representative; and
- (g) downloading into the personal computer of the sales representative the quotation with the pricing for each item.

* * * * *

20

25

30

35

40

45

50

55

60

65

An Approach to Reducing Delays in Recognizing Distributed Event Occurrences

Madalene Spezialetti
CSEE Department, Packard Lab
Lehigh University
Bethlehem, PA 18015

Abstract.

In order to aid in analyzing the activity of distributed computations, monitoring systems have been developed which allow a user to specify behavior of interest in the form of an event definition. The monitoring system then analyzes the behavior of the computation, notifying the user when the computation exhibits the behavior described by a definition. To accomplish such an analysis, the activity of the components of an event must be monitored individually and information pertaining to their values sent to a designated monitor for evaluation. Due to the need to accumulate state information, delays often result between the occurrence of activity specified by an event definition and the detection of the activity by the monitoring system. This paper presents an approach to reducing delays in event recognition by incorporating knowledge about the characteristics of the behavior to be recognized into the evaluation protocol. These characteristics are used in the development of techniques to localize information required for event recognition at the place of evaluation, thus decreasing the impact of communication delays on the recognition task and leading to a reduction in the time which elapses between an event's occurrence and its recognition.

1. INTRODUCTION

As distributed and parallel computing capabilities increase, there is a growing need to develop techniques which aid in analyzing the behavior of such systems. One approach to aid in this analysis allows users to describe activity of interest in the form of *event definitions* [1,2,5,8,10,11,13,15]. These definitions are predicates which test the states of various system elements. The activity of the elements specified by an

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1991 ACM 0-89791-457-0/91/0011/0155...\$1.50

event definition is monitored, and data pertaining to their activity collected for analysis. This data is organized into a view of the computation state, and the event definition is evaluated in terms of the states of its operands. An event *occurs* at the point that the behavior of the computation fulfills the specification of its definition, that is, at the point that the application of the definition's operators to the states of the operands would yield a TRUE result. An occurrence of the event is *recognized* at the point that the monitoring system detects its occurrence.

Due to the decentralized nature of distributed computations, the task of event recognition often requires the transmission of state information from a processor where a monitored element resides to another processor for evaluation. This communication can lead to an unpredictable delay between the occurrence of the activity specified in an event and the recognition of its occurrence. In many situations, such as process control and various real time applications, it is desirable to reduce the delay between event occurrence and recognition. In addition, the use of a centralized location for the processing and evaluation of event information can lead to message congestion, potentially delaying the event recognition further as well as possibly impeding the activity of the monitored processes.

A time lapse will always result between an event's occurrence and its recognition due to the need to evaluate the event. This delay can be minimized if the only delay between an event's occurrence and its recognition is that which results from the actual evaluation. Such a recognition will be referred to an *Immediate* recognition. If there is a delay, due to the need to accumulate additional data from other processors, between the point that the event occurs and its occurrence is recognized, the recognition will be referred to as a *Delayed* recognition.

This paper addresses the task of reducing the delay between an event's occurrence and its recognition by applying techniques which attempt to recognize an

event immediately. Techniques are presented which address the problem of monitor placement. The characteristics of an event are utilized to determine if an allocation of the monitoring task can be made which would guarantee the immediate recognition of the event's occurrence. If such an allocation can not be made, knowledge pertaining to the stability of the elements tested by an event are incorporated into analysis techniques in order to allow the re-use of information in event evaluations. These techniques attempt to localize information at the place of evaluation in order to reduce the impact of communication delays on the recognition of event occurrences.

The next section provides a description of events and of the various monitoring modules which cooperate to recognize events occurrences. Section 3 describes the event characteristics which will be utilized to aid in the reduction of recognition delays. Section 4 briefly discusses the characteristics of immediate recognitions. The task of determining if an immediate recognition can be guaranteed through the placement of monitors based on the examination of an event is addressed in Section 5. Section 6 presents a means of reducing the recognition delay by re-using information in successive evaluations based on the characteristics of the monitored elements. The integration of these techniques into a protocol for forming views of a distributed computation's state and examples demonstrating the application of the techniques are presented in Sections 7 and 8, followed by the conclusion in Section 9.

2. EVENTS AND MONITORS

An *event definition* is a description of the activity which is to be detected by a monitoring system. Specific definition languages which provide various forms of abstraction have been developed [1,2,5,10,11,13]. However, since this paper concentrates on general recognition problems, the existence of a general purpose specification capability, similar to that used in [1], is assumed. In particular, the existence of a set of *primitive events* is assumed, where primitive events are the (application specific) lowest level of system activity which may be used in the construction of event definitions. Intuitively, primitive events would be available to test such entities as communication transmission or reception, memory and file accesses, process or entity state and process status.

An event definition, is assumed to be a predicate whose operands, or *components*, are combined or tested using relational, logical or temporal operators. Events may be hierarchical in nature, so that components may be primitive events or events which were defined by the user (thus, references to components can also be applied to events and *vice versa*). It is also assumed that a set

of *event operators*, which take events as their operands, are available to test the value of the events which are their operands. The event operators are comprised of the set of logical operators {AND, OR} which test the truth value of the operands, and the set of temporal operators {!,#,',} which test the occurrence or order of occurrence of the operands. The semantics of the event operators are given in Figure 1.

LOGICAL OPERATORS

E1 AND E2 TRUE while both E1 and E2 are simultaneously TRUE.
E1 OR E2 TRUE while either E1 or E2 are TRUE.

TEMPORAL OPERATORS

E1 # E2 TRUE when both E1 and E2 have become TRUE.
E1 | E2 TRUE when either event E1 or event E2 becomes TRUE.
E1 [E2 TRUE when E2 becomes TRUE after E1 has become TRUE.

Figure 1. Event Operators.

In order to recognize an event occurrence, data regarding the states of the components must be collected and evaluated. Given the absence of a global clock or a single shared memory in a distributed system, some method must be provided by which the information can be organized into consistent representations of the system state [9]. The responsibility for maintaining a protocol to enable such a view of a distributed computation must also be integrated with the event recognition system. While the notion of a global clock or shared memory does not exist for a distributed system at large, such a notion is available on an individual processor within the system. The organization of monitoring modules and delegation of monitoring responsibilities utilizes this notion in order to reduce the overhead of defining consistent views of the system and to reduce the time required to collect the information pertaining to the current states of components located at a specific processor.

At each processor at which a component of a particular event is located is an *event monitor*. This monitor will maintain information pertaining to the states of each component of the event which is located at that processor. Associated with each process is a *component*

monitor, which is responsible for detecting changes to the monitored components within that process and, when an alteration occurs to a component, transmitting the value of that alteration to the appropriate event monitor. It is assumed that, when a change occurs to a monitored component, the operation of informing the event monitor of that change is atomic. In that way, the event monitor located at a particular processor maintains information regarding the current values of all of the event's components at that processor. Event monitors are also responsible for maintaining whatever inter-processor protocol is required in order to obtain a consistent view of the component states at various processors.

Evaluation monitors must also be designated for an event. An evaluation monitor may be assigned the task of evaluating an entire event or some portion of an event and is responsible for accumulating and organizing the data which is required to perform the evaluation and potentially make recognitions. An evaluation monitor may be placed at any processor, even one at which none of the components or elements associated with the monitor's event reside. However, such an arrangement would always necessitate some data transmission in order to perform an evaluation and therefore would always result in some delay, due to interprocessor communication, between an event's occurrence and its recognition. To reduce such a delay, evaluation monitors are more likely to be located at a processor at which some component of the event resides. In such a case, the evaluation responsibilities are incorporated into the event monitor at that processor. In that way, information pertaining to the components located at that processor is readily available for use in any evaluation. However, for generality, whenever discussing the assignment of evaluation responsibilities, the module will be referred to as an evaluation monitor, although it may serve as an event monitor as well.

3. EVENT CHARACTERISTICS

Given that the components of an event to be recognized will be scattered throughout a system, the information regarding the changes to their states must be communicated to the evaluation monitor or monitors responsible for recognizing the event occurrence. It is the communication of component state information to evaluation monitors which must be analyzed in order to determine if a recognition can be characterized as Immediate and it is this communication, and its impact on the monitoring task, which the techniques described in this paper attempt to reduce. The characteristics of the event to be recognized and of its individual components are utilized to achieve this reduction. Two characteristics are primarily utilized by these techniques: *the identification of critical components* and *component*

stability.

3.1. CRITICAL COMPONENTS

Many factors contribute to the determination of when an event occurrence can be recognized. Such factors include communication delays due to the need to accumulate monitoring information and the time required for event analysis and verification. However, while various considerations influence the recognition of an event, the ability to isolate the point at which the event becomes TRUE is a major factor in determining when an event occurrence can be recognized.

A change to the value of a single component, $\text{change}(c_i)$, or simultaneous changes to the value of two or more components of an event, $\text{change}(c_i, c_j, \dots)$, constitute a change in the value of the event itself. The *monitoring* of a computation in order to detect an occurrence of an event can be viewed as a series of $(\text{change}(e) :: \text{eval}(e))$ pairs, where each $\text{change}(e)$ is associated with either a change in the value of one component of e or simultaneous changes to two or more components of e :

$$(\text{change}_1(e) :: \text{eval}_1(e)), (\text{change}_2(e) :: \text{eval}_2(e)) \dots$$

where for $\text{change}_i, \text{change}_j$, $i < j$ implies change_i preceded change_j , under the consistent view of the system activity utilized by the monitoring system. A component c_k is a *critical component* of an event, e , if

$$\begin{aligned} \text{change}_i(c_k) :: \text{eval}_i(e) &= \text{TRUE} \\ \text{AND} \\ \text{change}_{i-1}(c_j) :: \text{eval}_{i-1}(e) &= \text{FALSE} \end{aligned}$$

that is, the critical component is the component whose change causes an occurrence of the event. Since the occurrence of the critical component signals the occurrence of the entire event, it follows that the recognition of the critical component's occurrence influences the recognition of the event. In particular, *the time lapse between the occurrence of an event and its recognition can be no less than the time lapse between the occurrence of the critical event and its recognition*. However, it should be noted that while it is beneficial to be able to identify the critical component of an event, it is not always possible.

3.2. STABILITY OF EVENT COMPONENTS

The second event characteristic which will be utilized to aid in the reduction of recognition time is *stability*, or *monotonicity*. Given that event components are dispersed throughout a system, the value of a component may change between the time that its value is sent to an evaluation monitor and the time that the value is used in

an evaluation. By recognizing the stable properties which may be exhibited by the components of an event, it is possible to infer the current state of a remote component's value. This knowledge can allow information regarding a component's state to be re-used, thus reducing the need to wait for current state data for each evaluation. Thus, the ability to guarantee the stability of component values affects the ability to reduce the time required to recognize an event occurrence. A component can be classified as monotonic, non-monotonic or dependent monotonic.

3.2.1. MONOTONICITY

A component is monotonic if, upon reaching a particular value, it retains that value from that time onwards. Let $eval_i(c)$ represent the value of a component c at time t . Component c is monotonic if there exists some holding value, h_v , such that if there exists some i , for which $eval_i(c) = h_v$, then for all $j, j > i$, $eval_j(c) = eval_i(c) = h_v$. In general, monotonic components test if a certain situation has ever occurred. For example, the event (*Process A accessed file F*) is monotonic since it will be TRUE at the point that file F was accessed by process P and will remain TRUE thereafter. An event is *true monotonic* if its holding value is TRUE. All references to monotonic events henceforth in this work are assumed to refer to true monotonic events.

A component which exhibits monotonic properties can be either intrinsically monotonic or operator monotonic. The intrinsic monotonicity of a component is determined at the primitive event level, that is, a primitive event which fulfills the monotonic criteria is intrinsically monotonic. In general, such primitive events test the occurrence of an action. Examples of intrinsically monotonic primitive events are (*process A reached line 1000*), (*process B wrote to file F*) and (*process C has terminated*). Since each of these primitive events test if an action has taken place, the event will become TRUE when the activity occurs, and will not revert back to FALSE.

For operator monotonic components, the operator for which the component is an operand determines its monotonicity. In particular, although a component may not be intrinsically monotonic, it may be associated with an operator which indicates an occurrence test. Although the component may be testing the current value of an object (an intrinsically non-monotonic characteristic), the operator causes the operand, and thus the expression, to become monotonic. Thus operands which, taken in isolation, are not monotonic can fulfill the monotonic criteria due to the operator with which they are associated. In particular, temporal operators, which test the order of occurrence of events, impart

monotonicity on their operands. That is, once the component has become TRUE, its occurrence would have fulfilled its contribution to the event. Even if the value of the component itself may later change, its contribution toward the event's recognition will remain stable. Further, since the operator imparts monotonicity on its operands, an expression formed with a temporal operator is monotonic as well, that is, once the specifications of its components, as they contribute to a particular recognition, have been fulfilled, the expression will become and remain TRUE.

Given the stability of monotonic components, the knowledge that such a component has reached its holding value can be utilized to reduce the amount of data which must be processed in order to recognize an event occurrence. In particular, once the holding value has been communicated to a monitor responsible for recognizing an event of which it is a part, that value can be used in repeated evaluations of the event until an occurrence of the event has been recognized. Since values which the evaluation monitor already possesses can be utilized in evaluations, the use of monotonic information can lessen the delay between the occurrence of an event and its recognition. In the best case, all but one of the monitored components will remain in the protocol, reducing the task of recognizing a distributed event to a local recognition of its critical component's occurrence, thus eliminating any delay due to inter-processor communication between the event occurrence and its recognition.

3.2.2. NON-MONOTONICITY

Components which do not exhibit the stable properties of monotonicity are termed non-monotonic. A non-monotonic component is a component whose value can change at any time. No assertions can be made with respect to the length of time for which a non-monotonic component will maintain its current value nor can any projections be made as to its future value. For example, the event ($A.x < 3$), which tests the value of a variable x at a process A , is non-monotonic. As the value of $A.x$ changes, the evaluation of ($A.x < 3$) will also change to reflect the values assumed by $A.x$. Thus, based on the current value of the evaluation, no predictions can be made as to the future values which ($A.x < 3$) may assume. Events which contain non-monotonic components are more difficult to recognize, and their recognition tends to be far more communication intensive. Since the value of the events may change at any time, a greater need arises to inform other monitors of the fluctuations in their values. Further, given the instability of a non-monotonic component, it is not possible, in general, for a monitor to be assured that the current value of a remote component corresponds to the

information which the monitor possesses.

3.2.3. DEPENDENT MONOTONICITY

While non-monotonic components are volatile in nature, there is a bridge between monotonic and non-monotonic components, namely, dependent monotonic components. Dependent monotonic components are components which are non-monotonic by nature but which exhibit monotonic characteristics over certain known periods of time, referred to as *holding spans*. During the holding span, the component will maintain a holding value. Dependent monotonicity enables the efficiencies which can be gained through monotonicity to be utilized in recognizing events whose components are inherently non-monotonic. For any dependent monotonic component, a dependent element can be identified. The dependent element is that entity which determines the delineation of the holding span, that is, that element which controls the period during which the component will maintain its holding value. A dependent element could, for example, be a synchronization element, such as a semaphore, an activity which involves inherent synchronization, such as synchronous communication, or a module or element to which control is explicitly transferred, such as module which is invoked via a remote procedure call (RPC) facility [3].

Unlike monotonic or non-monotonic components, which can be classified by inspection, the determination of an event's status as dependent monotonic can be determined either by the nature of the component itself, by the activity of the process associated with the component or by the environment in which the component's event can occur. A component is explicitly dependent monotonic if its dependent monotonicity can be identified via the examination of the event. Such components imply an explicit transfer of control or suspension of processing by their nature. An example in explicitly dependent monotonicity is the event (*A suspended on semaphore S*). The event itself implies a suspension of activity of process A and identifies the dependent element, S, which will control the period over which this event would remain in the holding value of TRUE.

A component is implicitly dependent monotonic if it is non-monotonic in nature, but it assumes monotonic characteristics due to the behavior of a computation. The dependent monotonicity of such a component can not be determined by the inspection of the definition. Rather such factors as the transfer of control, or access to a synchronization or locking facility by the process at which such a component resides, indirectly causes the component to enter a holding span. Thus, it is necessary to examine the control flow and synchronization mechanisms utilized by the component processes in order to determine if a component is implicitly

dependent monotonic and the holding spans over which the component behaves in that manner. As an example, consider the event $((A.x < 3) \text{ AND } (B.y > 5))$, in which both of the components are non-monotonic. If during the course of the computation, process A makes a remote procedure call to process B, the component $(A.x < 3)$ can be guaranteed not to change, that is, it will behave monotonically, from the time that the call is made until control is returned to A.

4. REDUCTIONS IN RECOGNITION TIME

The recognition of an event occurrence involves the collection of required component information, the organization of this information into consistent views of the computation's activity, and the evaluation of the event definition based on the component values. The approaches to the reduction of recognition time presented in this paper attempt to minimize the impact of communication delays on the evaluation procedure. If the impact of communication delays on the recognition of an event can be totally eliminated, then the recognition is termed *Immediate*. That is, the of an event, e , is characterized as Immediate if there is no delay, due to monitoring related communication, between the occurrence of an event and its recognition. Thus, a recognition is Immediate if it meets the following criteria:

- 1) The evaluation monitor responsible for recognizing the occurrence of e is located at the same processor as the critical component of e .
- 2) All state information required to recognize the occurrence of e is available at the evaluation monitor at the point that the change occurs to the critical component which, in turn, constitutes an occurrence of the event e .

The techniques presented in this paper to reduce recognition delays attempt to accomplish Immediate recognitions of events. Although such a recognition can not always be achieved, the techniques can still lead to a reduction in the time lapse between an event's occurrence and its recognition. These approaches focus on two aspects of the recognition task: the placement of the evaluation monitor relative to critical components and accumulation and the utilization of component information.

5. EVALUATION MONITOR PLACEMENT

In order to recognize an event occurrence Immediately, evaluation monitors which are capable of recognizing an event occurrence must be located at the critical components of the event. Thus, the task of monitor placement attempts to assign evaluation responsibilities so that they may be located at the critical components,

leading to a decentralization of the evaluation task. The first step in determining if a placement of evaluation monitors is possible which would permit an Immediate recognition involves the analysis of event operators. In particular, for each operator, a determination is made as to whether or not, due to the semantics of the operator, it is possible to identify a critical component for that expression. In this analysis, a definition will be viewed as an expression tree. Initially, each leaf node represents a component, and each interior node represents an operator, and thus a potential point of evaluation. A link between a parent node, p , and a child, c , indicates that data must be transmitted from the processor where c is located to the processor where p is located in order for p to perform an evaluation. The placement of evaluation monitors is represented by grouping an operation with one or more of its children, indicating that an evaluation monitor responsible for performing the specified operation will be placed at the same processor as the child with which it is grouped. The tree is modified using the following rules to group interior nodes p , and their children c_1 and c_2 :

- PR1** If, due to the semantics of the operation represented by p , both c_1 and c_2 can be designated critical components, p is *split*, such that the operator represented by p is grouped with both c_1 and c_2 , and p' , the parent of p , becomes the parent of both c_1 and c_2 . The splitting of an evaluation node indicates that an instance of the evaluation monitor associated with p is placed at the processors where c_1 and c_2 are located, thus insuring that the evaluation monitor is placed at the critical components.
- PR2** If, due to the semantics of the operation associated with p , one child, c_i , can be designated a critical component for the expression, then the child c_i is *absorbed* into p , that is c_i is associated with the parent node, p , and the node formerly associated with c_i is pruned.
- PR3** If due to the semantics of the operation associated with p , neither c_1 nor c_2 can be designated a critical component for the expression, then one of the children is absorbed into the parent node p . In this situation other criteria, such as the location of other components of the event or the desire to centralize or decentralize the evaluation, may be utilized to determine which child of p should be absorbed. While the choice of which child to absorb may effect the amount of monitoring messages actually needed to recognize the event, since the critical component can not be determined via examination of the event, the choice of either component does not effect the decision as to whether or not an Immediate recognition can be

guaranteed.

The above rules are iteratively applied until there are no remaining operators which have not been associated with one of the processors at which a basic component is located. An Immediate recognition of the event e can be guaranteed by applying the placement of evaluation monitors as indicated by the transformed expression tree if either:

- 1) All links have been removed from the transformed expression tree or
- 2) For every group of nodes connected by links, all nodes of the group are located at the same processor.

If the above criteria are met, then it can be guaranteed that, for each potential critical component of an event, all information required to recognize an occurrence of the event would be present at the evaluation monitor located at the same processor as the critical component at the time of the critical components occurrence. Thus, at any time that a change should take place to a critical component such that its new value would constitute an occurrence of the event, it would be possible to recognize the event occurrence based solely on information available to the evaluation monitor located at the critical component at that time.

Let us consider the application of the above algorithm, based on the operators described above, to the expression $c_1 \text{ op } c_2$. An expression formed using one of the operators {!,OR, or}, will become TRUE when *either* of its operands becomes TRUE. Since both components individually constitute known critical components, PR1 is applied to these operators. The operator {'}, implies that c_2 occurs after c_1 and therefore will always be the critical component. Thus, PR2 would be applied to the operator {'}. The states of both components determine the value of an expression formed with any of the operators {AND,#,and,<,>,<=,>=,<>}. Thus, PR3 would be applied to these operators.

Consider the event $((B \text{ in critical_sec}) \text{ OR } (C \text{ in critical_sec})) \& (A \text{ blocked on receive}) / (D \text{ in critical_sec})$, where processes A and B are located at processor P_1 , process C is located at processor P_2 and process D is located at processor P_3 . It is assumed that the critical section tests are made by local program counter checks, thus the values of all components are determined based on the status of the individual processes. The events are represented in the figure using the process names only. The initial expression tree is shown in Figure 2a. The OR operator is split between its operands, resulting in the configuration in 2b. The AND operator however can not be split, and therefore is

grouped with one of its children, (*A blocked on receive*), as shown in 2c. Finally, operator *I* is split between its children, resulting in the final tree in 2d. Since a link exists between (*B in critical_sec*) and (*A blocked on receive*), and these components are not located at the same processor, an Immediate recognition of this event can not be guaranteed. However the application of the algorithm indicates a placement of monitors which will localize the evaluation of components where possible, to decrease the amount of state information transmission and thus increase the likelihood of an Immediate recognition when possible.

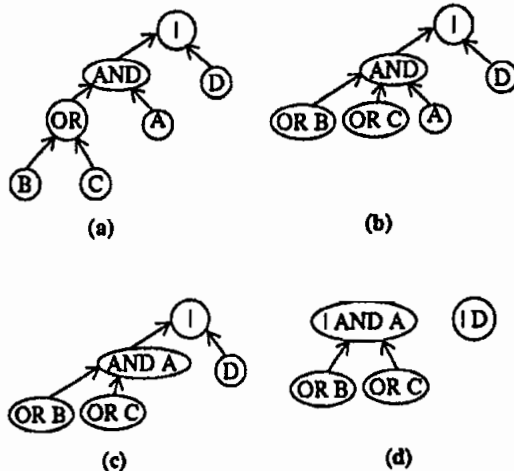


Figure 2: Grouping of Components and Operators

It should be noted that even if a transformed expression tree does not meet the above criteria, an Immediate recognition of the event's occurrence is not necessarily precluded. Due to the order in which the monitored activity occurs and the availability of component information, Immediate recognitions of the event may occur. However, the inability to meet the criteria does imply that such a recognition can not be guaranteed in advance based solely on the semantics of the event. The next section presents techniques which attempt to achieve Immediate recognitions on-the-fly as a means of reducing the time lapse between an event's occurrence and its recognition in situations where an Immediate recognition can not be guaranteed in advance. These techniques utilize the knowledge of components' monotonicity to permit the re-use of information which an evaluation monitor may possess.

6. ON-THE-FLY IMMEDIATE RECOGNITIONS

In the absence of a global clock, it is not generally possible to construct a view of the system state as it existed at a specific point in time. Thus, in order to evaluate the state of a computation for the purposes of

event recognition, some method must be employed by which the component state information can be formed into a consistent representation of the system state. Techniques have been developed which provide such a view of system activity [4,6,7,9,12,14,16]. Thus, for current purposes, it will be assumed that the monitoring system is provided with a series of consistent views, v_1, v_2, \dots, v_n , of the monitored components' states. These views are constructed at the evaluation monitor based on the reception of a value for each component of an event which is tagged with a view number, indicating the view of which the component value is a part. All component values of a particular view are considered to be "concurrent".

To reduce the time lapse between the occurrence of an event and its recognition, the evaluation monitor maintains information regarding the monotonicity of components whose state information it possesses so as to allow the re-use of the information in subsequent evaluations. *Current_view_number*, indicates the view in which the current evaluation of component information is taking place and the view in which an event occurrence will be recognized if the definition evaluates to TRUE. The use of information in the evaluation of an event e for a particular view is determined based on the following rules:

- EV1 If the value of a component c_x is received for a view v_j , such that c_x is either inherently monotonic or operator monotonic and its value, designated $c_{x,j}$, is the holding value of the component, the value $c_{x,j}$ can be used in the evaluations of any subsequent views $v_k, k \geq j$.
- EV2 If the a component c_x is dependent monotonic and known by the evaluation monitor to be within a holding span, the last value assumed by c_x before entering the holding span, designated value $c_{x,j}$, can be utilized as the current value of c_x in an evaluation of $v_{\text{current_view}}$ until it is known that c_x is no longer in its holding span.
- EV3 If a component c_x is non-monotonic, then only its value $c_{x,\text{current_view}}$ can be utilized in the evaluation for $v_{\text{current_view}}$.
- EV4 If sufficient information has been received to evaluate an expression of e for $v_{\text{current_view}}$, the value of the expression will be utilized for the evaluation in $v_{\text{current_view}}$ according to the monotonic characteristics of its operands, that is, according to (EV1) if monotonic, according to (EV2) if dependent monotonic and according to (EV3) if non-monotonic.

A recognition for view $v_{current_view}$ can be characterized as Immediate if the evaluation monitor for the event is located at the same processor as the critical component of the occurrence and, at the time the value of the critical component for view $v_{current_view}$ is received, all information required for the evaluation of e for $v_{current_view}$ is present at the evaluation monitor.

7. EXAMPLE INTEGRATION OF ON-THE-FLY EVALUATION TECHNIQUES WITH A VIEW PROTOCOL

The above techniques can be applied to any protocol which is utilized to define consistent views of a computation in order to perform event recognitions. However, for the purposes of discussion, we will assume that *simultaneous regions* are utilized to define such views. Briefly, the technique introduces messages which are sent between an event's monitors in order to define numbered regions. When a change occurs to a monitored component in a region, r , which may result in an occurrence of an event, the component's event monitor is informed of the change. The current values of the components associated with the event monitor are associated with the region number r and the state information is sent to the appropriate evaluation monitor tagged with region number r . Marker messages tagged r are then sent to the other event monitors for the event to signify that a change occurred to a component in region r . Finally, the current region number of the event monitor at which the component whose value changed is located is incremented to $r+1$.

When an event monitor whose components are executing in a region r' receives a region message marked r such that $r' \leq r$, the event monitor carries out same steps as are executed if one of its components had experienced a change. If, however, $r' > r$, the message is ignored, since the relationship indicates that the receiving event monitor had already processed its component information for region r and progressed into a subsequent region. Information contained in like-numbered regions is considered simultaneous. Since simultaneous region numbers are maintained by the event monitors, all monitored components at a given processor execute in the same region at any given time. A complete description of the techniques can be found in [14,17].

In order to perform on-the-fly Immediate recognitions, information pertaining to monotonicity must be integrated into the scheme which is used to construct the state views. In particular, the evaluation monitors must be provided with information so that they can deduce if any of the components are monotonic and have reached their holding values, or are dependent monotonic and within a holding span at the time of the evaluation. An

evaluation monitor can determine if any components which are local to it are exhibiting monotonic characteristics at the time of the evaluation. However, some means must be provided to communicate this information to evaluation monitors for those components which are not local to the evaluation monitor.

First, let us consider the handling of monotonic components. An evaluation monitor can determine that a component is monotonic and has reached its holding value if either (a) the component is inherently monotonic and the component is known to the evaluation monitor or (b) the component is operator monotonic and the evaluation monitor performs the operation. However, due to the assignment of evaluation responsibilities or the hierarchical definition of an event, an evaluation monitor may not have direct access to knowledge regarding the monotonicity of a particular component. In such a case, a *holding token*, $(component_id, region_number)$, will be generated by the monitor which has access to the information indicating that such a monotonic component has reached its holding value. The first field of the token identifies the component and the second field indicates the region number in which the component reached its holding value. Thus, either through information available locally to an evaluation monitor, or due to the possession of a holding token for a particular component, an evaluation monitor can be provided with sufficient information to determine when a monotonic component has reached a holding value and the region in which the value was assumed. This information is then utilized to determine the applicability of (EV1) of the evaluation algorithm.

To aid in reducing the delay in event recognition, information regarding dependent monotonicity must also be available at the evaluation monitor. Since no other process will be executing at the processor where an evaluation monitor is executing at the time that the monitor is executing, any component located at the same processor where an evaluation monitor is located can be considered dependent monotonic with respect to the activity of the evaluation monitor. However, information must also be available regarding the dependent monotonicity of components located at other processors. In order for an evaluation monitor to be assured that a component is dependent monotonic and within its holding span at the time of evaluation, the dependent element for that component must be located at the same processor as the evaluation monitor. As a means of providing an evaluation monitor with such knowledge, *control tokens* are used. A control token, $\langle process_id, region_number \rangle$, is associated with each process at which a monitored component resides and is utilized to indicate the locus of control of a process. The first field of the token identifies the process with which it is asso-

ciated and the second field indicates the number of the last region in which the process was active. The control token reflects the control status of a process P according to the following rules:

- CT1 If process P is active, the control token of P is located at the processor where P is located.
- CT2 If process P is suspended as a result of an explicit transfer of control (directly or indirectly) to process P' , such that P can only resume execution as a result of an explicit return of control (directly or indirectly) from process P' , then the control token for P is held at the processor where P' is located.
- CT3 If process P is suspended as a result of an operation which does not meet the specifications of CT2, then the control token of P is held at the processor where P is located.

Control tokens located at any processor are assumed to be visible to any evaluation monitor located at that processor. The location of the control token for a process P at a processor indicates that the dependent element for event components related to P is located at that processor. Thus, control tokens provide an evaluation monitor with sufficient information to determine both *explicit* and *implicit* dependent monotonicity. By examining the control tokens located at its processor, an evaluation monitor can determine if any of its components are dependent monotonic and within a holding span at the time of the evaluation.

7.1. EXAMPLES OF ON-THE-FLY RECOGNITIONS

As an example of the application of the on-the-fly evaluation techniques as integrated with view construction via simultaneous regions, consider the activity shown in Figure 3, which depicts the recognition of the event $E = (A.x < B.y)$, that is an event which tests the relationship between the values of an element x at process A and element y at process B. It is assumed that the processes reside at different processors. It is also assumed that a rendezvous communication protocol is utilized by the processes A and B. Since it can not be determined, based on the evaluation of the event, which component's change would result in an occurrence of the event, the evaluation will be carried out at the processor where B is located. The evaluation responsibilities for the "less than" test will be incorporated with the event monitor which maintains the state and simultaneous region protocol for B.y. For clarity, only messages which carry information for evaluation are shown. While messages related to the establishment of the simultaneous regions have been omitted, horizon-

tal markings at each process indicate where the region boundaries were formed.

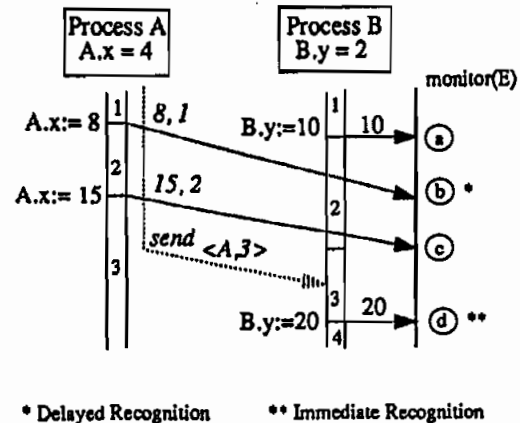


Figure 3. Activity while monitoring $E = (A.x < B.y)$.

At the point at which the activity is shown, the value of A.x was 4 in region 0 and the value of B.y was 2. It is assumed that the evaluation monitor has already received this information. First, A.x is assigned the value 8. A.x's new value is associated with region number 1 and the value is sent to the evaluation monitor tagged with the region number. Before A's message reaches the evaluation monitor, B.y is assigned the value 10. This value is transmitted directly to the evaluation monitor. (Region numbering is handled at the monitor, since it also serves as B.y's event monitor.) At the point that B.y's value of 10 for region 1 is received, an evaluation can not be performed since A.x's value of 8 for region 1 has not yet been received. When this information is subsequently received, the evaluation ($8 < 10$) is performed and an occurrence of the event is recognized. Although the conditions of the event were actually fulfilled at the point that B.y was assigned the value 10, the absence of A.x's state information for region 1 caused a delay between the event occurrence and its recognition.

Another change then occurs to A.x when it is assigned the value 15. This value is captured in region 2 and is again transmitted for evaluation. When this

value is received, B.y's value for region 1 is also captured for region 2. The resulting evaluation ($15 < 10$) yields a value of FALSE, and thus no recognition for region 2 is made by the monitor. Process A then performs a synchronous send to process B. Since A will suspend until B performs a receive, A's control token, marked with the region in which the process suspended, is also transmitted via a monitoring message. (The visibility of this token to an evaluation monitor implies that the state of A, and therefore the value of A.x, would remain at the value which it possessed in the region preceding the transfer of control.) B.y is then assigned the value 20 which is transmitted to the monitor and B.y's new value is associated with region 3. At the point that B.y's value is received by the evaluation monitor, the control token for A is visible to it, guaranteeing that the value of A.x is dependent monotonic and within a holding span. Thus, the value of A.x for region 2 is utilized for the evaluation in region 3. The evaluation ($15 < 20$) is performed and an occurrence of the event is recognized. Since B.y was the critical component for the event occurrence in region 3, and the value of A.x available to the evaluation monitor located at B's processor could be utilized, an Immediate recognition of the event was achieved.

In the previous example, information indicating that a remote component was dependent monotonic and within a holding span enabled a reduction in recognition time. Consider now, the recognition of the event $E3 = ((E1 \text{ AND } (E2 \# (C.y < 25)))$. Where event $E1 = (A.i > 10) / (A.j > 50)$ and event $E2 = (B.x = 7)$. The responsibility for evaluating the event E3 is assigned to a monitor located at the same processor as C. The event E1 is completely local to A and thus is evaluated and recognized at A. Similarly, the evaluation of E2 is carried out at the processor where B is located. Only the results of these evaluations are transmitted to the evaluation monitor at C. Neither the actual components nor the characteristics of the components of E1 and E2 are known to the monitor at C. The initial values of the individual components are shown in Figure 4. As in the previous example, it is assumed that the values of the components for region 0 are known to the evaluation monitor at this point in the computation and have already been processed.

First, A.j assumes the value 51. This change results in an occurrence of E1, thus causing the value TRUE to be sent to the evaluation monitor for E3 along with the region number, 1, in which it occurred. In addition, E1 is formed using the temporal operator I, causing the components of E1, as well as the event E1, to be operator monotonic. That is, since the event tests an occurrence of its components, the result of TRUE from the evaluation will remain stable. Thus, in addition to

$$E1 = (A.i > 10) / (A.j > 50) \quad E2 = (B.x = 7)$$

$$E3 = E1 \text{ AND } (E2 \# (C.y < 25))$$

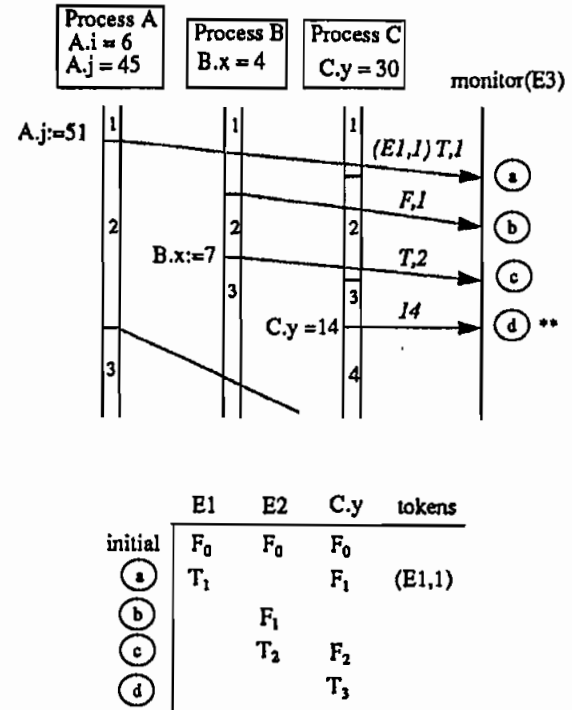


Figure 4. System Activity When Monitoring E3.

the value for region 1, a holding token, (E1,1) is also transmitted to the evaluation monitor of E3. The reception of the state information for region 1 causes region 1 to be established for C.y. The change to E1 also causes the value of E2 to be captured for region 1 (although the region protocol messages have been omitted) and its value is sent to the evaluation monitor of E3. Based on the component values for region 1, E3 did not occur and thus there is no recognition.

B.x is then assigned the value 7. This change causes an occurrence and recognition of E2, thus the value of TRUE is transmitted to the evaluation monitor of E3 for region 2. Since, from the perspective of the monitor at B, event E2 is non-monotonic, no holding token is transmitted. The reception of the message also causes the value of C.y to be associated with region 2. In addition, since a holding token for E1 for region 1 is available, its state value for region 1 can be utilized in an evaluation, eliminating the need to wait for E1's

value for region 2. Thus, an evaluation of E3 is made upon the reception of the message for E2 and found to be FALSE. Finally, an assignment of the value 14 to C.y is made and captured in region 3. This change causes the expression $(C.y < 25)$ to be fulfilled. In addition, since E2 is associated with the operator # and the evaluation of that operator is carried out at the evaluation monitor for E3, the evaluation monitor can determine that E2 is operator monotonic with respect to event E3. Thus, E2's value of TRUE for region 2 can be utilized immediately in an evaluation, as can E1's value of TRUE for region 1. The evaluation yields a result of TRUE and thus a recognition of E3 is made for region 3. In addition, since the change to C.y was the critical component for this occurrence and all state information needed to make the recognition was present at the evaluation monitor located at C.y when the change occurred, an Immediate recognition was possible.

8. CONCLUSION

Given the need to accumulate data for the analysis of a distributed computation's activity there can be a considerable time lapse between the occurrence of monitored activity and the analysis of information pertaining to its occurrence. This paper described a two-pronged approach to reducing this delay through the use of techniques which attempt to achieve Immediate event recognitions, that is, which attempt to eliminate any delay, due to inter-processor communication, between an event's occurrence and its recognition by the monitoring system. First, a technique was presented to determine if an Immediate recognition of an event's occurrence could be guaranteed via the placement of evaluation monitors based on the characteristics of an event's operators. Second, techniques were presented which incorporated the stability of an event's components into a strategy for the re-use of component state information. These techniques enable data local to an evaluation monitor to be used in successive evaluations of an event, thus contributing to the reduction of the time required to make a recognition and increasing the possibility that a particular event occurrence can be recognized Immediately.

REFERENCES

- [1] P.C. Bates and J.C. Wileden, "High-Level Debugging of Distributed Systems: The Behavioral Abstraction Approach," *Journal of Systems and Software*, vol. 3, no. 4, pp. 255-264, 1983.
- [2] P.C. Bates, "Debugging Heterogeneous Distributed Systems Using Event-Based Models of Behavior," *Proceedings of Workshop on Parallel and Distri-*

buted Debugging, pp. 11-22, 1988.

- [3] A. Birrel and B. Nelson, "Implementing Remote Procedure Calls", *TOCS*, vol.2, no.1, pp. 39-59, 1984.
- [4] K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Transactions on Computer Systems*, vol. 3, no. 1. pp. 63-75, 1985.
- [5] R. Curtis and L. Whittie, "Bugnet: A Debugging System for Parallel Programming Environments," *Proceedings of the 3rd International Conference on Distributed Computing Systems*, pp. 67-78, 1982.
- [6] C. Fidge, "Partial Orders for Parallel Debugging", *Proceedings of Workshop on Parallel and Distributed Debugging*, pp. 183-194, 1988.
- [7] D. Haban and W. Weigal, "Global Events and Global Breakpoints in Distributed Systems," *Proceedings of 21st Hawaii International Conference of System Sciences*, pp. 166-175, 1988.
- [8] P.K. Harter, D. Heimbigner, and R. King, "IDD: An Interactive Distributed Debugger", *Proceedings of the 5th International Conference on Distributed Computing Systems*, 1985.
- [9] L. Lamport, "Time, Clocks and Ordering of Events in Distributed Systems", *Communications of the ACM*, vol. 21, no. 7, pp. 558-565, July 1978.
- [10] R.J. LeBlanc and A.D. Robbins, "Event-Driven Monitoring of Distributed Programs," *Proceedings of the 5th International Conference on Distributed Computing Systems*, pp. 515-522, 1985.
- [11] C.C. Lin and R. LeBlanc, "Event-based Debugging of Object/Action Programs", *Proceedings of Workshop on Parallel and Distributed Debugging*, pp. 23-33, 1988.
- [12] B.P. Miller and J.D. Choi, "Breakpoints and Halting in Distributed Systems", *Proceedings of the 8th International Conference on Distributed Computing Systems*, pp. 316-323, 1988.
- [13] E.T. Smith, "Debugging Tools for Message Based, Communicating Processes," *Proceedings of the 4th International Conference on Distributed Computing Systems*, pp. 303-310, 1984.
- [14] M. Spezialetti and J.P. Kearns, "Efficient Global Snapshots," *Proceedings of the 6th International Conference on Distributed Computing Systems*, pp. 382-388, 1986.
- [15] M. Spezialetti and J.P. Kearns, "A General Approach to Recognizing Event Occurrences in Distributed Computations", *Proceedings of the 8th*

International Conference on Distributed Computing Systems, pp. 300-307, 1988.

- [16] M. Spezialetti and J.P. Kearns, "Simultaneous Regions: An Approach to the Consistent Monitoring of Distributed Computations for Event Occurrences," *Proceedings of the 9th International Conference on Distributed Computing Systems*, pp. 61-69, 1989.
- [17] M. Spezialetti, "A Generalized Approach to Monitoring Distributed Computations for Event Occurrences", Ph.D. Dissertation, University of Pittsburgh, 1989.